New Approaches for Modeling and Reliability Assessment of Infrastructure Flow Networks

A Dissertation
Presented to
The Academic Faculty

by

Yanjie Tong

In Partial Fulfillment
of the Requirements for the Degree Doctor of Philosophy
in the
Civil and Environmental Engineering

Georgia Institute of Technology
April 2021

COPYRIGHT © 2021 BY YANJIE TONG

New Approaches for Modeling and Reliability Assessment of Infrastructure Flow Networks

Approved by:

Dr. Iris Tien, Advisor
School of Civil and Environmental Engineering
Georgia Institute of Technology

Dr. Nagi Z Gebraeel
School of Industrial and Systems Engineering
Georgia Institute of Technology

Dr. John E Taylor
School of Civil and Environmental Engineering
Georgia Institute of Technology

Dr. Yao Xie
School of Industrial and Systems Engineering
Georgia Institute of Technology

Dr. Samuel Coogan
School of Civil and Environmental Engineering
Georgia Institute of Technology

Date Approved: [May, 2021]

ACKNOWLEDGEMENTS

I would like to give special thanks to my parents who support me in the decisions I make to pursue a Doctoral of Philosophy. The efforts and sacrifices they made in my early years form my courage and readiness to face the challenges in future. My PhD career will be impossible without their guidance and help.

I would also like to show my appreciation to my advisor Dr. Iris Tien, who patiently guide me through the requirements to be a qualified PhD student. When I began my first year as a PhD student in 2015, I was panic and excited at the same time. I have no idea of where my research is going, what references or prior works should I study, what is the correct form of a publishable paper, etc. Dr. Tien encourages me with inspiring advices when I am confused with my research directions and corrects me patiently when the submitting paper is not in a proper form.

For my peers, Ruhui Chen, Dan Li, Ke Liu, Xinjun Dong, Xi Liu, Chloe Applegate, Yijian (Albert) Zhang, Ajay Saini, Cynthia Lee, Jorge-Mario Lozano, Paola Vargas, life will be difficult and boring without your company. Insightful advises at group meeting motivates my steps in making contributions. Playing basketball and hiking enhance my physical wellness to handle the incoming challenges in life.

Time flies in the process of pursing a PhD degree in Civil and Environmental engineering. It is an unforgettable journey of both joys and sorrows. 6 years of research teach me how to balance the life and work as well as how to handle the pressure when the expectations are not met.

LIST OF TABLES

LIST OF FIGURES

SUMMARY

Infrastructure flow networks, such as water pipelines, power distributions, etc., play an important role in our daily life. These networks are subject to increasing threats and disruptions, both from natural disaster events and more targeted attacks. It is critical to be able to accurately assess the reliability of a network to inform decisions regarding maintenance, retrofitting, replacement, etc., to secure the functionality of the network. Infrastructure flow networks are also complex with many components and links operating together. Thus, computational tractability and efficiency is paramount. In this thesis, we propose new approaches for modeling and reliability assessment of infrastructure networks, investigating the reliability in terms of both connectivity and flow capacity. Connectivity, indicating whether the receiving ends or destination points are reachable, is the most fundamental requirement of a network. Flow capacity, which represents the efficiency and level of performance of a network, is another important factor in reliability assessment.

The definition of reliability can be generally described as a relationship between demand and capacity. A precise analysis on reliability requires accurate estimations on both parts. In this thesis, we predict the upcoming demand on the network by analyzing collected time series data. The objective is to predict values characterizing the nodes in a network. We propose a pairwise gated recurrent unit (Pairwise-GRU) approach to analyze the collected time series data at each node. By considering the influence from the neighboring nodes in addition to historical data, we improve both the accuracy and confidence level of the prediction.

The capacity of a network, in this thesis, is explored in two aspects: connectivity and flow capacity. For connectivity, we propose the probability propagation method (PrPm) and directed probability propagation method (dPrPm). These methods originate from the idea of belief propagation in graphical models, instead propagating a probability distribution to result in reliability assessments at all terminal nodes in the network. Both PrPm and dPrPm work for multiple-source-multiple-sink networks with significantly reduced time complexity in computational efficiency compared to existing approaches. PrPm applies to general networks with approximated solutions. dPrPm provides upper bounds and lower bounds for acyclic directed networks.

For flow capacity, we propose algorithms to conduct multistate Bayesian network (BN) inference and a modified theory of maximum flow to deal with the multiple-source-to-multiple-sink scenario. In the multistate BN inference, we tackle the challenge of computational intractability for multistate BN by introducing a compression algorithm and corresponding inference algorithm for the network information. Significant decreases in previously prohibitive storage requirements are found with some tradeoff in computational efficiency. In the modified maximum flow work, we extend the traditional single-source-to-single-sink theory of maximum flow to the multiple-source-to-multiple-sink scenario with the ability to implement constraints and limit the feasible routes across the network.

In sum, we present new methods to investigate the reliability of infrastructure flow networks in terms of connectivity and flow capacity. We include a Pairwise-GRU approach to improve the credibility of the inputs to the network. Two different methods are then proposed for both

connectivity and flow capacity analysis. Over a range of applicable scenarios and networks, the methods are shown to advance current reliability assessment capabilities with increased accuracy and computational efficiency.

Chapter 1. Introduction

1.1 Motivation

Individuals and societies are highly dependent on the services provided by infrastructure flow networks, such as water pipelines, power distributions and transportation networks. Infrastructure flow networks are playing an important role in our daily life by securing our safety, convenience as well as the functionality of society. Precise and timely approaches for modeling and reliability assessment of infrastructure flow networks measure the functionality of the network and enable the analysis of the deterioration system performance due to hazards of increasing frequency and severity. These analyses facilitate decision making relating to retrofitting, replacing, maintenance, etc. to ensured continued operation of infrastructure flow networks under varying scenarios.

Reliability can be measured as a function between capacity and demand. For different subjects, the definition of reliability and the focus of investigation can be different. For example, when we look at the reliability of a building structure, we are interested in the stated conditions like deformation, settlement, cracking, etc. If we change the subject to infrastructure flow networks, two of the major concerns are connectivity and flow capacity.

1.2 Data collection and inputs

The accuracy of reliability analyses depends on the credibility of input data. The first step in conducting the reliability analysis of an infrastructure flow network is collecting data from the

field. The field data helps researchers to have a better understanding of potential demands on the network. By analyzing the historical data, researchers can make credible predictions on the future demands on the network. The conduct of reliability assessment is defining the relationship between demand and capacity. As a result, the accuracy of the input data, which relates to the demand of the network, influences the outcome of any reliability analysis.

When the standard of capacity does not meet the desired demand, a network is considered as a failure or unreliable in a reliability assessment, and vice versa. Capacity is determined by the intrinsic characteristic of the network, depending on the material, connection, environment, etc. In this thesis, we measure the capacity in two different ways: connectivity and flow capacity. Demand, taken as a time-dependent factor in this thesis, highly relies on the human activities. In this thesis, we focus on improving the accuracy and uncertainty of time series data prediction of demand based on collected historical data across a network.

In Chapter 2, a more accurate and confident time series prediction in this thesis is proposed based on a recurrent neural network, i.e., gated recurrent unit (GRU) approach. For a more accurate and confident outcome, we consider not only the history of the time series data but also its interaction with its neighboring node. We call this a Pairwise-GRU approach. The characteristics of the GRU, such as updated gate and reset gate, is preserved. We modify the structure of the GRU by adding an additional layer connecting two neighboring nodes in the network. The performance of the proposed approach, in terms of accuracy and uncertainty, is tested by an electricity network in the southeastern U.S.

1.3 Connectivity

The most fundamental requirement of an infrastructure flow network is that it stays connected, and the receiving ends or destination points are reachable. Thus, connectivity is one of the basic indicators in reliability assessment of infrastructure flow networks. The infrastructure flow network is modeled as a graph, which consist of nodes and links, delivering resources from source nodes to sink nodes. The reliability of the network, in terms of connectivity, is determined by the uncertainty in the functionality of nodes and links to deliver resources to end points in the network. However, reliability analysis of the connectivity of complex networks is often limited by large and exponentially increasing computational requirements with system size, especially when the probabilities of rare events are of interest. To address this challenge, in this thesis, we propose two methods: the probability propagation method (PrPm) and directed probability propagation method (dPrPm).

In Chapter 3, the probability propagation method (PrPm) is introduced to find the reliability of infrastructure flow network of general interest. It provides an approximated analytical solution with reduced computational requirements, reducing computation cost from an exponential increment with system size to polynomial increment. The target network works for both directed graphs and undirected graphs with one source node and multiple sink nodes. The idea of PrPm originates from the belief propagation for inference in graph theory, which passes and updates the message between nodes in the network. In our proposed PrPm, the message is defined as a pairwise nodal distribution. At each propagation step, we update the pairwise nodal distribution based on the connection between the nodes. In the theoretical deduction part in Chapter 3, we find that the assumption overestimates and underestimates at the same time, which balances out

the error to some extent. When the sink nodes are reached, an approximated solution regarding connectivity of the network is defined. To demonstrate the performance of PrPm, three test applications are introduced: a seven-component network, a power distribution network, and a general grid network.

The performance is enhanced when we narrow down our target network to acyclic directed graphs. In Chapter 4, following the similar idea shown in PrPm, we put forward a new analytical probability propagation method, named the directed probability propagation method (dPrPm). Like the message passing mechanics in PrPm, dPrPm passes the message, which also refers to pairwise nodal distribution, to the neighboring nodes through a link-adding sequence. Based on the assumptions made in derivation of the updating rules, in dPrPm, we find the upper bound and lower bound of all node reliabilities in connectivity. The computational time also reduces from a typical exponential increment to polynomial increment. Three test applications are shown to demonstrate the performance of dPrPm: directed grid network, power distribution network, and gas pipeline network.

1.4 Flow Capacity

The other topic relating to the reliability of infrastructure flow networks is flow capacity, which is a higher standard than connectivity. Due to attacks from natural hazards and the aging and degradation of materials, the flow capacity of a network is damaged resulting in a potential reduced response to demand. The network is composed of individual components. The flow capacity of the network can be described as an underlying function relating to the performance of individual components. In this thesis, we investigate the relationship between network flow

capacity and individual component performance in two separate ways: one through the use of Bayesian network (BNs) and one through a graphical solution to solve a maximum flow problem.

In Chapter 5, we use Bayesian networks (BNs) to model the dependencies between individual components and system performance. The results given by a BN are helpful for decision makers in system maintenance, repair, and replacement management with the ability for real-time updating of network performance assessments based on new information. Compared to the more widely studied binary BN, in this thesis, the state of components defined is extended to multistate. One of the major challenges in utilizing BN modeling for infrastructure flow networks is the exponential increases in computational cost as a function of the number of components in the network. We tackle this challenge by introducing a lossless compression algorithm for all the intermediate factors created during the variable elimination process. As a trade-off for computational time, a significant compression rate is found in the typically prohibitive storage requirement for BNs. For test applications, a seven-component network and its variational form are used to demonstrate the performance of the proposed algorithm.

In Chapter 6, we analyze the flow capacity of the infrastructure flow network in another way – through a graphical solution to solve the maximum flow problem. We modify the traditional theory of maximum flow in graph theory by expanding the discussion from the single-source-to-single-sink scenario to the multiple-sources-to-multiple-sinks scenario. To account for monetary and temporal considerations in real networks, we also put limitations, e.g., on travel cost and travel time, on acceptable routes between origin-destination pairs. We begin with the idea of

augmenting paths in traditional maximum flow theory. However, in this thesis, we separate the augmenting paths into two cases to extend the application to multiple-sources-multiple-sinks networks. We also provide a quasi-optimization solution to recommend recovery strategies after a hazard event or disruption. The performance of proposed framework is tested against grid networks of different sizes for comparison and the real-world Western U.S. Double-Stack Container Network.

1.5 Organization of this thesis

The organization of this thesis is summarized as the following: Chapter 1is the introduction of the thesis. Chapter 2 focuses on the data preparation. Chapter 3 and Chapter 4 focus on the connectivity aspect of the reliability assessment. Chapter 5 and 6 focus on assessing network reliability in terms of flow capacity. From Chapter 2 to Chapter 6, all five chapters begin with a literature review of previous work in the area. A theoretical deduction of the proposed method is then described. Each of these chapters ends with test applications to demonstrate the contributions and advances of the proposed method. An overview of the thesis is shown in Figure 1.1, including a brief summary of previous work, contributions, and example networks analyzed for each proposed method.

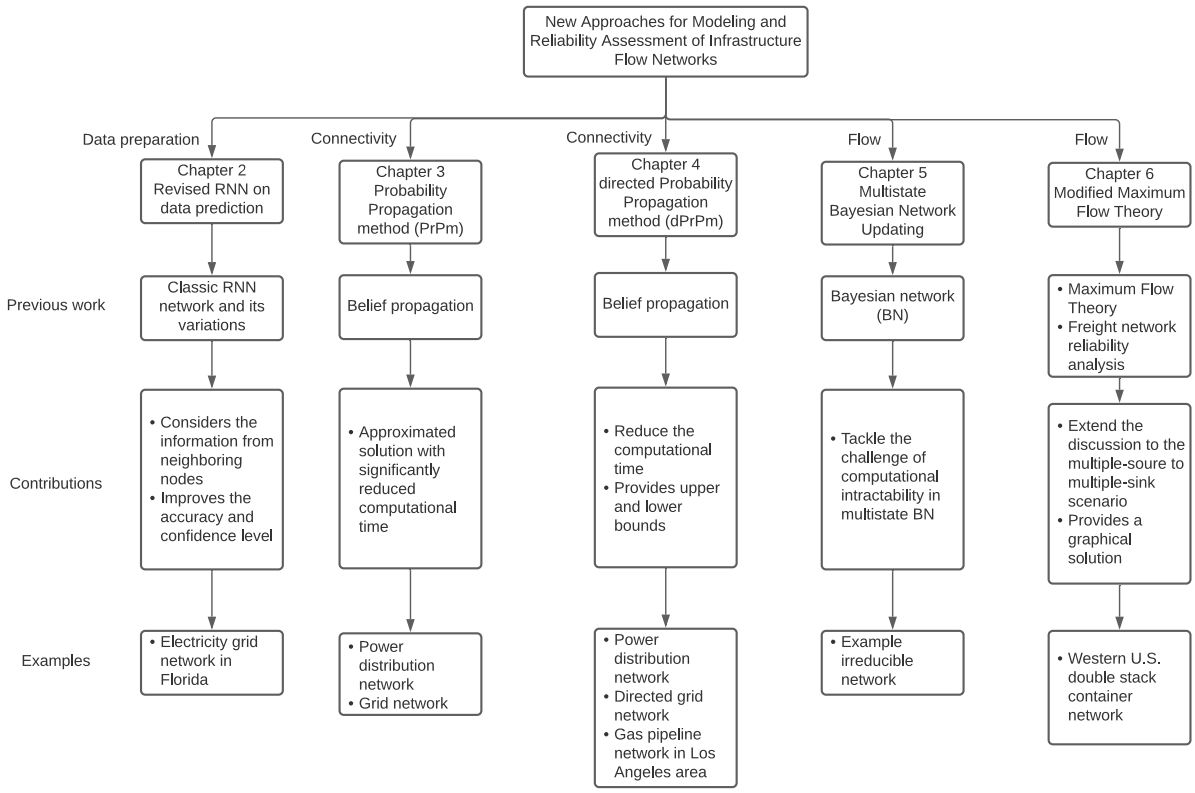

Figure 1.1 Overview of thesis

Chapter 2. Data preparation

The structure of this chapter can be summarized as the following. First, we make a literature review of existing methods on time series prediction other than RNNs. Then, we propose our modified model based on Gated Recurrent Unit (GRU), named as Pairwise-GRU. After that, a real-life application on an electricity network is tested on our proposed Pairwise-GRU. Finally, the performance of our proposed model and effects of including neighboring nodes are analyzed. The contribution of this chapter is concluded.

2.1 Introduction – RNN and its variations

A recurrent neural network (RNN) processes a temporal sequence through a directed graph. Its applications include handwriting recognition, speech recognition, and time series prediction (Frank et al., 2001). In Frank et al. (2001), time series prediction by neural network is applied on Lorenz data, voice traffic demand and tree ring data. Discussions are made on the selection of the size of a sliding window. In this paper, we focus on the structure of hidden units in neural network. The simplest form of a RNN is shown in Figure 2.1.1, where inputs and outputs are connected by the hyperbolic tangent transformation. The performance of naïve RNNs is often limited by gradient vanishing and long-term recognition because of the limited number of parameters (Li et al., 2018).

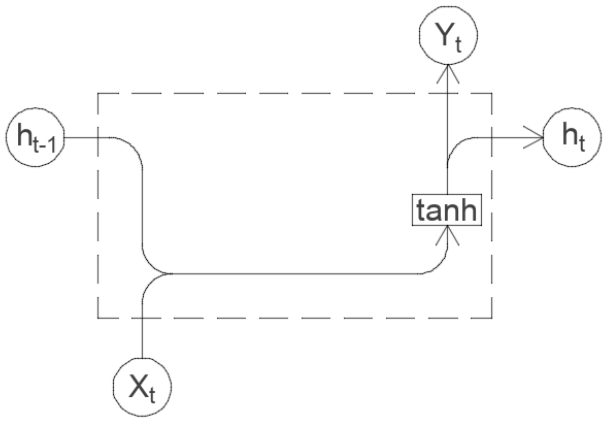

Figure 2.1.1 Naïve RNN

Thus, to avoid the long-term dependency problem and to accommodate lags between data points particularly in time series prediction problems, the Long Short-Term Memory network (LSTM) was developed by Hochreiter and Schmidhuber (1997). Compared to the naïve RNN in Figure 2.1.1, an additional layer of cell states, denoted as C_t in Figure 2.1.2, is introduced. A typical LSTM network is shown in Figure 2.1.2, which is also featured by a chain like structure. In both Figure 2.1.1 and Figure 2.1.2, the complete Naïve and LSTM RNN are repetitions of Figure 2.1.1(2.1.2), connecting heads to tails. A LSTM can be improved by reducing the number of parameters, facilitating the computational efficiency.

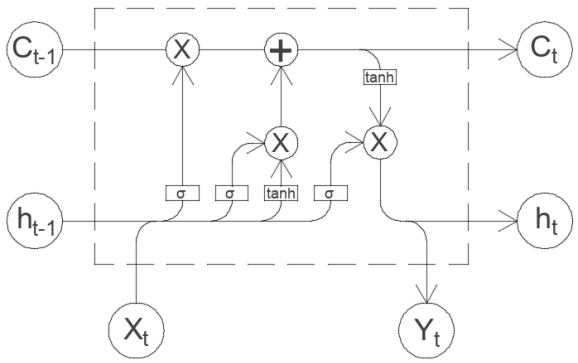

Figure 2.1.2 LSTM RNN

For improved computational efficiency, a variational form of LSTM – gated recurrent units (GRUs) – was introduced by Cho et al. (2014). The structure of a GRU is shown in Figure 2.1.3. GRUs solve the long-term dependency problem and gradient vanishing problem with fewer parameters than LSTMs. In this paper, the proposed Pairwise-GRU is based on the traditional GRU, connecting two GRUs with a transition box in between. By doing so, the approach is able to take into account the effect of neighboring nodes in time series prediction.

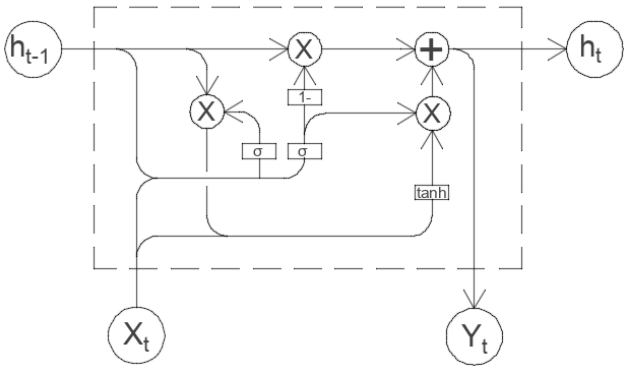

Figure 2.1.3 Gate Recurrent Unit (GRU)

The rest of the chapter is organized as follows. The next section describes previous work in time series prediction, including limitations of existing approaches. Then, the proposed approach for time series prediction in nodal networks that is based on GRUs, called Pairwise-GRU, is described. The Pairwise-GRU approach is then tested on a real-world application of an electricity network. The performance of the proposed approach and the effects of including neighboring nodes are analyzed. The contributions of this chapter are then concluded.

2.2 Literature review – other time series prediction related methods

Aside from RNN and its variational forms, there are also many different ways to make a time series prediction. In the following paragraph, we focus on three different methods: Relevance vector machine, Group Method of Data Handling and Grey System.

Relevance vector machine (RVM) is first proposed by Tipping (2001) featuring the sparse Bayesian learning. Similar to the support vector machine (SVM), RVM has a kernel function and the parameter learning is based on Bayes rules. Based on RVM, a multi-scale relevance vector regression approach is proposed by Bai et al. (2014) to forecast the daily urban water demand. The performance of RVM depends on the choice of bandwidth in kernel function, which may lead to overfitting problems. SVM can be applied to time series predictions as well. In Sapankevych and Sankar (2009), SVM is applied in various scenarios such as financial market forecasting and control system process. Due to the highly nonlinear aspect of data, similar to the case of RVM, the choice of kernel function largely influences the performance of prediction.

Group method of data handling (GMDH) is a family of inductive algorithms for computer-based mathematical modeling of multi-parametric datasets that features fully automatic structural and parametric optimization of models. The auto layer-generation stops when a preset criterion is met. In Nikolaev and Iba (2003) and Shelekhova (1995), time series are break down into harmonic forms, where the harmonic model parameters are learnt through GMDH. Since there is no limitations on the number of layers that can be generated, the overfitting problem exists in GMDH as well.

Deng (1982) came up with a Grey system dealing with incomplete information. A grey system means that a system in which part of information is known and part of information is unknown. With this definition, information quantity and quality form a continuum from a total lack of information to complete information – from black through grey to white. Applications of Grey system are found in predictions of economic (Kayacan et al., 2010) and traffic volume (Xu and Zhang, 2010) growth. However, when the underlying governing equation fails to depict the growth rules, the prediction becomes unreliable.

In Wei et al. (2021), RNN is used for time series prediction on pore water pressure. Comparison of the performance of time series prediction are made over RNN, LSTM and GRU. To mitigate the problem of overfitting, a dropout technique following a Bernoulli distribution is used. The structure of hidden units in RNN is preserved, which cannot include the influence from neighboring nodes. Researches are made on the structure of hidden unites in RNN to improve the performance of time series prediction. In Hu and Zheng (2019), additional operators are added to capture the short-term dependencies in dataset. Likewise, in this paper, we created an addition layer between two GRUs to reflect the influence of neighboring nodes.

Another way to address the gradient explosion/vanishing problem is by setting up a window size on the training data. In Bai et al. (2014), prediction on the daily urban water demand is limited to a training window size of 7 days/1 week. In Frank et al. (2001), the optimum window size is chosen by trial and error. The optimal window size is determined by auto-correlation method and the saturated correlation dimension method, Holzfuss and Mayer-Kress (1986). For simplicity, in

the test application part of this paper, window size for prediction is selected as 48, which accounts for the nearest 48 hour dataset.

2.3 Theoretical deductions

The network of interest consists of nodes with time series data information at each node. The goal is to make a prediction at each node based on the given historical data. To account for the neighboring effect in time series prediction, we use the proposed Pairwise-GRU to include the information from neighboring nodes.

2.3.1 Pairwise GRU

The proposed method originates from the traditional GRU. The structure of proposed Pairwise-GRU is shown in Figure 2.3.1.1. It can be decomposed into 2 separated GRU connected by the box in the middle. The box in the middle ensures that one node can share the information with its neighboring node. By adding the middle box, the influence of neighboring nodes is considered.

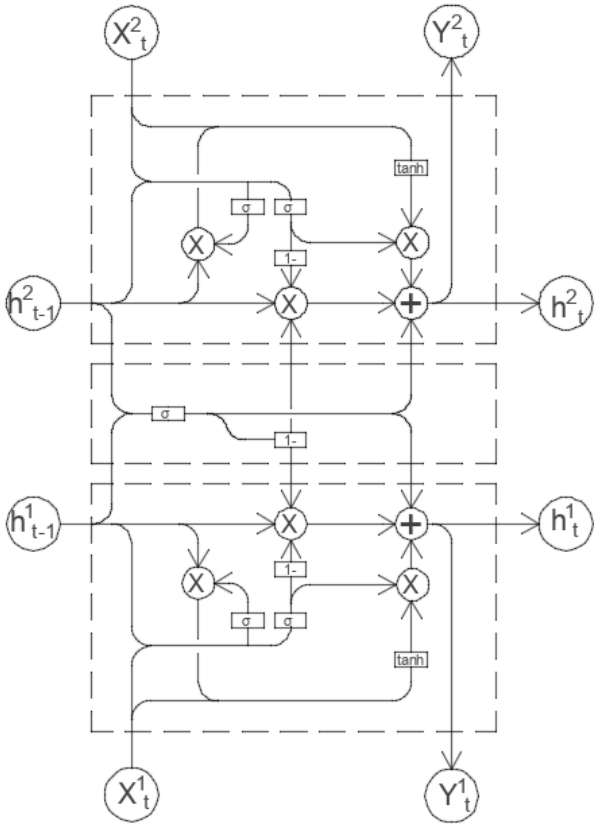

Figure 2.3.1.1 Propose Pairwise-GRU

The calculation process is listed as below:

$$z_t^1 = \sigma(w_z^1 \cdot [h_{t-1}^1, X_t^1]) \cdots\cdots (1)$$

$$z_t^2 = \sigma(w_z^2 \cdot [h_{t-1}^2, X_t^2]) \cdots\cdots (2)$$

$$c_t = \sigma[w_c \cdot [h_{t-1}^1, h_{t-1}^2]] \cdots\cdots (3)$$

$$r_t^1 = \sigma(w_r^1 \cdot [h_{t-1}^1, X_t^1]) \cdots\cdots (4)$$

$$r_t^2 = \sigma(w_r^2 \cdot [h_{t-1}^2, X_t^2]) \cdots\cdots (5)$$

$$\widetilde{h_t^1} = \tanh(w^1 \cdot [r_t^1 \otimes h_{t-1}^1, X_t^1]) \cdots\cdots (6)$$

$$\widetilde{h_t^2} = \tanh(w^2 \cdot [r_t^2 \otimes h_{t-1}^2, X_t^2]) \cdots\cdots (7)$$

$$h_t^1 = (1 - z_t^1) \otimes (1 - c_t) \otimes h_{t-1}^1 + z_t^1 \otimes c_t \otimes \widetilde{h_t^1} \cdots\cdots (8)$$

$$h_t^2 = (1 - z_t^2) \otimes (1 - c_t) \otimes h_{t-1}^2 + z_t^2 \otimes c_t \otimes \widetilde{h_t^2} \cdots\cdots (9)$$

$$Y_t^1 = h_t^1 \cdots\cdots (10)$$

$$Y_t^2 = h_t^2 \cdots\cdots (11)$$

2.3.2 Preprocessing

The performance of the prediction also depends on the training data input. In practice, we find that GRU performs better when the input is monotonically increasing. Also, when picking up training data for parameter learning, the recent historic data has larger effect on the prediction. Thus, we choose a window size of 48 hours and transform the data series into monotonic increasing data series. Let the training data series be X_i and transformed data series be $\widehat{X_\iota}$, where $i = 1,2, \ldots ,48$. The relation between $\widehat{X_\iota}$ and X_i is established as following,

$$\widehat{X_\iota} = \frac{X_i}{a_i \max(\{X_i\})}$$

where $\max(\{X_i\})$ refers to the maximum element in set $\{X_i\}$ and a_i is linearly decreasing scale factor ranging from 4.5 to 1.5. By this transformation, the original input training time series data is likely to be transformed into a monotonically increasing time series data ranging from 0 to 1.

2.3.3 Neighbor definition

There are numerous ways to define neighbors. For example, we can define the neighbors through physical connection. However, by the rules of thumb, we find the best way, in terms of computational efficiency and prediction performance, to define a neighboring node is based on the similarity check of time series data. Iglesias and Kastner (2013) and Gonzalez-Abril (2014)

propose multiple methods to identify the distance (similarity) between to two time series data. By these standards given in two papers above, if data series A is the closest (most similar) one to data series B, then node A is defined as the neighboring node of node B.

2.3.4 Learning

We define the loss function as the sum of squared error. The efficiency of learning depends on the initial value settings on the parameters. Thus, the learning process is divided into two parts. First, we learn the parameters for two traditional GRU separately. Then, we taken the parameter value learnt in the previous two separate GRUs as the initial value in the Pairwise-GRU. The learning process in Pairwise-GRU is by trial and error.

2.3.5 Testing

In the preprocessing, we transform the original input data into a monotonic increasing data series. Thus, once we produce an outcome, we need to undo the transformation to get the prediction. Let the outcome be Y and the prediction be $\hat{Y}$, the transformation from $\hat{Y}$ to Y is:

$$\hat{Y} = Y * a_y * \max(\{X_i\}),$$

where a_y is the scale factor depending on the prediction steps.

2.3.6 Evaluation

The performance of the proposed Pairwise-GRU is evaluated in two categories: accuracy and uncertainty. The accuracy of the prediction, p_j, is evaluated by comparing the predicted number

to the actual collected number. Let the predicted value be $\widehat{Y}_j$ and the actual collected number be $\overline{Y}_j$, precision at time step j will be:

$$p_j = 1 - |\frac{\overline{Y}_j - \widehat{Y}_j}{\overline{Y}_j}|$$

Coefficient of Variation (CoV) is used to measure the confidence level/uncertainty. CoV for predictions at time step j, σ_j, is determined by the performance of prediction in training set.

In sum, workflow of the proposed method can be summarized as shown in Figure 2.3.6.1.

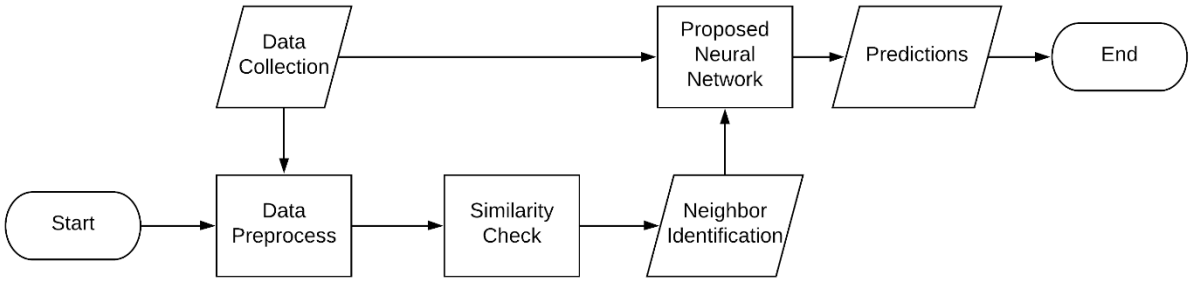

Figure 2.3.6.1 Workflow for building the proposed Recurrent Neural Network

2.4 Test applications – electricity grid network in Florida

Tests are completed on a 16 GB RAM computer in MATLAB_R2017b. Electricity grid network in Florida is take as the example network.

2.4.1 Electricity grid network in Florida

The electricity grid network in Florida is taken as the example network. Electricity consumption data is collected on an hourly base from December 16, 2019, to December 31, 2019, for a total of 361 consecutive time series datapoints collected at each station. This time period is chosen to

include data from normal operating as well as hazard conditions. In Figure 2.4.1.1, network configuration is extracted from the satellite map, where each station is treated as a node and the physical connections between nodes are kept as links.

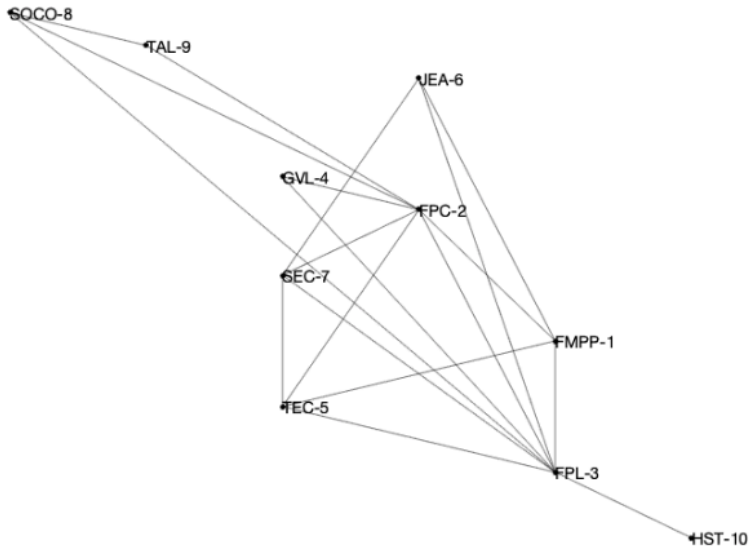

Figure 2.4.1.1 Network configuration

Based on the discussion above, the closest (most similar) data sets are considered as neighbors. After calculating the similarities between all pairs, the neighbors in this test application are defined as follow in Table 2.4.1.1. Judging from the results in Table 2.4.1.1, the highest similarity score has no relationship with physical connection.

Table 2.4.1.1. List of neighbors

Node of interest	Neighboring node
FMPP-1	TEC-5
FPC-2	TEC-5
FPL-3	FPC-2

Table 2.4.1.1 continued

GVL-4	SOCO-8
TEC-5	FPC-2
JEA-6	TAL-9
SEC-7	FPL-3
SOCO-8	GVL-4
TAL-9	JEA-6
HST-10	SOCO-8

Five methods (Grey System, GMDH, RVM, GRU and Pairwise-GRU) are used to compare the performance of prediction as shown in Figure 2.4.1.2. Predictions are made from time step 200 to 240. Precisions on the system level are calculated from 1-step-ahead prediction to 24-step-ahead prediction, i.e., 1-hour-ahead to 1-day-ahead predictions. Based on the curves on Figure 2.4.1.2, for short-term predictions, they are doing equally well. However, for the long-term prediction, the GRU and proposed Pairwise-GRU approaches perform better than the other three, with the proposed Pairwise-GRU approach showing the highest precision.

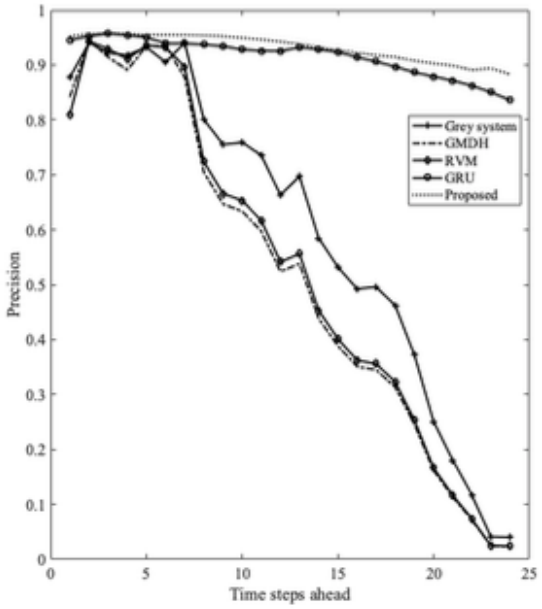

Figure 2.4.1.2 Methods comparison

Given GRU and Pairwise-GRU as the top performing methods, this section performs a detailed comparison between the two approaches under three cases, including both normal operating and hazard conditions. Comparisons are all made based on a 3-step-ahead prediction. Looking chronologically through the dataset, the three cases are as follows: from time step 140 to 180, a flooding hazard impacted node FPL-3 and node HST-10; for time step 200 to 240 and time step 300 to 340, these are considered as operating under normal conditions. GRU includes information from the time history only. The proposed Pairwise-GRU takes into account both the time history and information from neighboring nodes. The results comparing GRU with Pairwise-GRU are summarized in Table 2.4.1.2. From Table 2.4.1.2, across normal and hazard conditions, the Pairwise-GRU approach results in increased precision and decreased uncertainty (CoV) in the prediction compared to GRU. The improvement is more pronounced in the flood

hazard condition, which is an anomalous condition that is not accounted for when considering the time history only in the traditional GRU approach.

Table 2.4.1.2 Performance comparison on the system level

Condition	Time range	History only (GRU)		Proposed (Pairwise-GRU)	
		Precision	Average CoV	Precision	Average CoV
Flood	140-180	93.70%	0.0801	94.53%	0.0664
Normal	200-240	95.71%	0.0646	95.81%	0.0545
Normal	300-340	94.19%	0.0812	94.81%	0.0734

To compare results at each node, predicted results from GRU and Pairwise-GRU are shown in Figure 2.4.1.3, 2.4.1.4 and 2.4.1.5, where the bounds are obtained based on CoV and 95% confidence level. The confidence level is obtained through the performance of the training set. Assuming the prediction follows a normal distribution, we find the CoV of training set and apply the same CoV value to the predictions. The three time periods considered are the same as investigated in Table 2.4.1.2. The upper (lower) bound 1 are the results from GRU. The upper (lower) bound 2 are the results from Pairwise-GRU. For all stations, it can be easily seen that a narrower bandwidth is achieved by the proposed Pairwise-GRU approach, which considers the effect of neighbor nodes on the time series prediction.

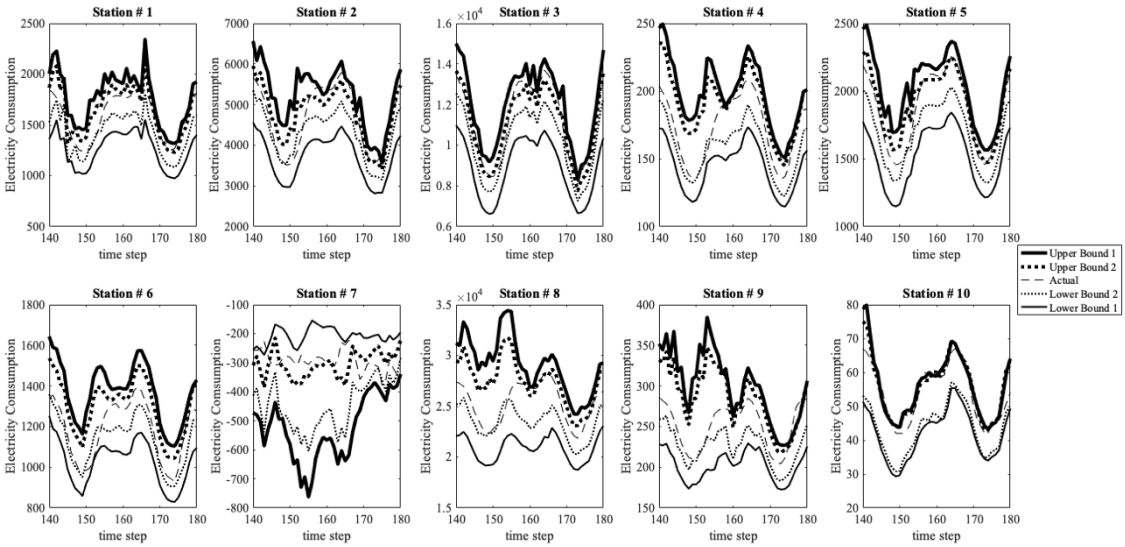

Figure 2.4.1.3 Time step 140-180

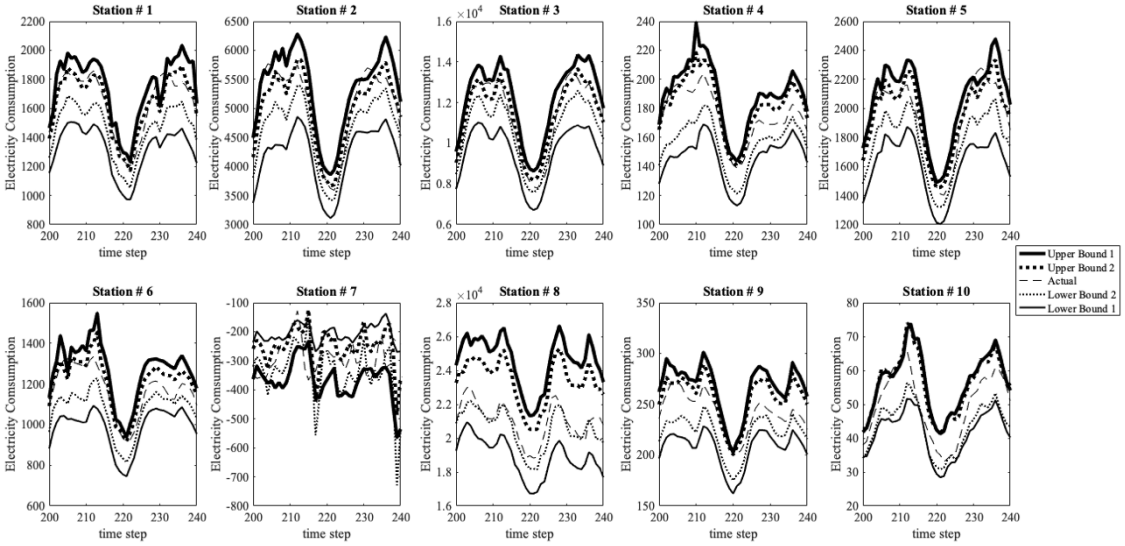

Figure 2.4.1.4 Time step 200-240

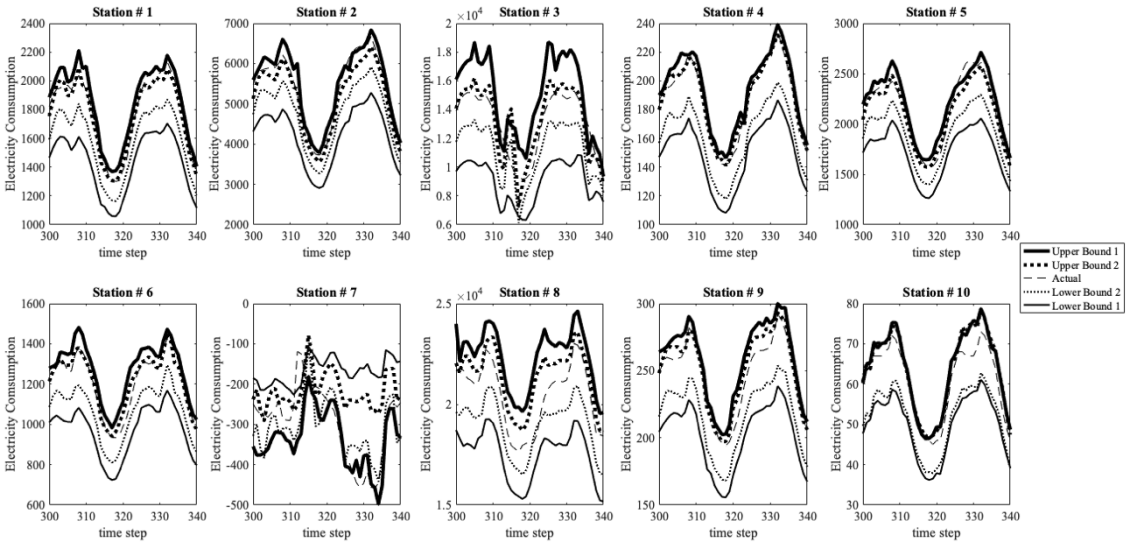

Figure 2.4.1.5 Time step 300-340

We also investigate the influence of neighboring node when the data collection among all stations are not synchronized, as shown in Figure 2.4.1.6. For the x-axis on Figure 2.4.1.6, 0 means data collection is synchronized on the node of interest and the neighboring node. A positive number on x-axis, e.g. +3, means data collection on the neighboring node is 3-hour ahead more up to date than the node of interest. Likewise, for a negative number, e.g. -3, it means the update on the neighboring node is 3-hour behind compared to what the node of interest has. We range the time difference in collection from -6 to 6. For each value, we sample 100 points on 3-hour ahead prediction, recording and plotting the results on Figure 2.4.1.6.

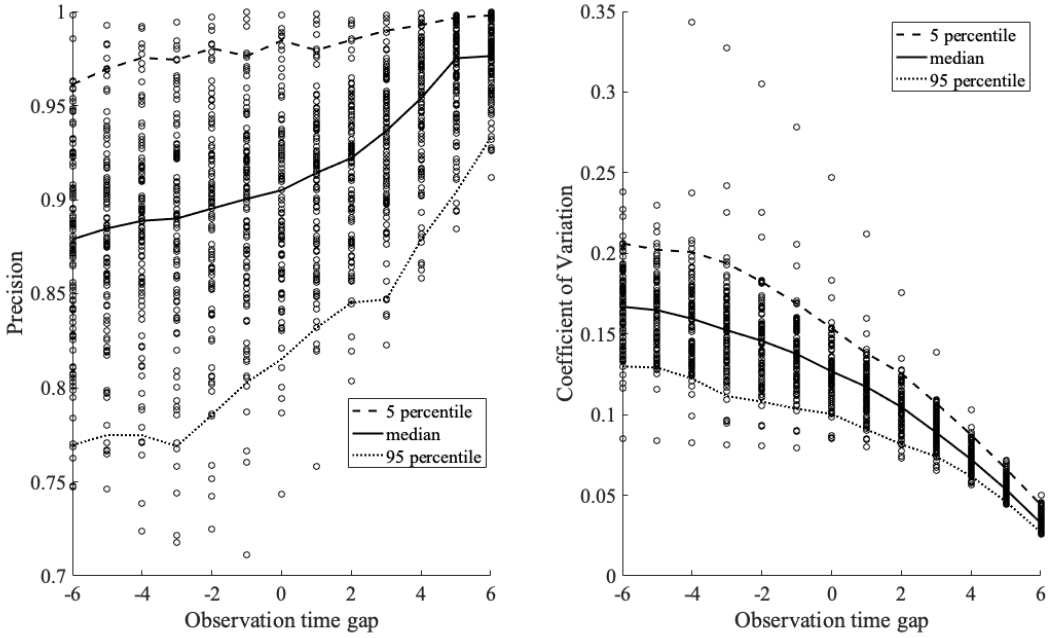

Figure 2.4.1.6 Influence of neighboring nodes

Both the median value cure and the percentile curve show a clear trend that we will benefit from considering the neighboring nodes when the neighboring node offers more information/ more up to date, increasing the accuracy and decrease the uncertainty. However, when the neighboring node is out of date compared to the node of interest, the performance of prediction is harmed by the neighboring effect.

For computational efficiency concerns, we also compare the computational time required for GRU and Pairwise-GRU. At the cost of time increment at less than 4 secs, we can potentially increase the accuracy and decrease the uncertainty of the prediction from Figure 2.4.1.6.

Table 2.4.1.3 Computational time comparison

	GRU	Pairwise-GRU
Computational time (sec)	11.55	15.38

2.5 Contributions

The contribution of this paper can be mainly summarized into four parts:

1. Based on the structure of traditional GRU, we propose a new Recurrent Neural Network, Pairwise-GRU, which builds the connection between neighboring nodes.
2. Despite of the fact that double parameters are involved in newly proposed recurrent neural network than the traditional network, the computational time/efficiency is not increased significantly because of the initial value setting in parameter learning.
3. Compare to the traditional GRU, the proposed Pairwise-GRU improves the accuracy and confidence level of the prediction slightly.
4. Since the proposed Pairwise-GRU takes in the information from neighboring nodes, the performance of the prediction improves significantly when the neighboring node has up-to-date information.

Chapter 3 Connectivity – Probability Propagation method (PrPm)

The rest of this chapter is organized as follows. First, an introduction and literature review of reliability analysis on general complex network. In the following section, a detailed description of Probability Propagation method (PrPm) is provided. Three test applications are used to demonstrate the performance of PrPm – a seven-component network, a power distribution network and a general grid network. Conclusions are made by the performance of the test application in terms of computation time and accuracy. The proposed method (PrPm) has been published by Tong and Tien (2019).

3.1 Introduction – reliability analysis on general complex networks

The reliability analysis of systems is important to assess and predict the performance of general complex networks. Furthermore, inference over the network enables identification of critical components in the system to support decision makers in setting inspection, maintenance, or replacement policies. Many approaches exist to assess the reliability of systems. These can generally be categorized as analytical or simulation-based. Analytical approaches often require computationally intensive total enumeration, either of the states of the components of a system or of its link or cut sets. These processes result in exact assessments of system reliability; however, they are typically characterized by exponential increases in computational cost with system size. As an alternative, simulation-based methods can be used. These result in approximations of the reliability with increasing the number of sample points generally yielding more accurate approximations of the exact solution. However, for large complex networks, generating a sample

point and determining its outcome is time consuming. Several methods to increase efficiency in sampling as well as to generate unbiased sample points have been developed.

In this chapter, we propose an alternative analytical method, called the Probability Propagation method (PrPm), to achieve accurate and computationally tractable reliability assessments of complex networks. The systems of interest consist of connected components modeled as networks of links and nodes. PrPm originates from the idea of belief propagation to pass messages from node to node. The passing of an approximated joint probability distribution results in an analytical solution for the system reliability. The accuracy of the resulting approximated solution is influenced by the assumptions made in message propagation. However, the computation time is reduced significantly from an exponential to a quartic increase with system size. In applying the proposed method to three test examples, the performance of PrPm is investigated compared to existing methods.

3.2 Literature review – reliability analysis methods

A brief description of existing methods for system reliability analysis is now provided, as well as the background for the proposed method. This is intended to provide an overview of methods for network reliability assessment rather than to serve as a comprehensive list. The reader is referred to texts such as Birolini (2004) for more details on reliability engineering.

At a fundamental level, systems can be assessed as a combination of the two basic network configurations: parallel and series. These configurations can be used to model redundancy and linear connections between components, respectively. Reliability analysis for simple networks is

easily determined by the characteristics of parallel and series systems. However, most realistic systems are in complex configurations, e.g. critical infrastructure flow networks such as power distribution networks with multiple sources and system interconnects, or overlapping pipeline designs for water and gas networks, which cannot be reduced to simple series and parallel configurations.

One method to analytically assess the reliability of general complex networks is through total enumeration, which lists all possible combinations of components and their corresponding outcomes in the system. Criticism for total enumeration comes from its exponential increase in computational cost as the number of components in the system increases.

An alternative analytical approach is based on the recursive decomposition algorithm (RDA) as presented in Dotson and Gobien (1979) and described in Lim and Song (2012). Selective RDA is proposed to improve the efficiency of the original RDA by identifying the most reliable paths. While the number of disjoint sets and computational cost is reduced heavily in the test network, a rigorous proof of faster convergence compared to using the shortest path is not provided. In cases where the most reliable paths are not significantly more dominant than others, the computational cost may still follow an exponential increase. In Kim and Kang (2013), the authors extend the application of RDA from one initial and one terminal node to general multi-initial and multi-terminal node networks. In RDA, certain components in a link set are considered to be failed in a graph. Graphs are then decomposed into sub-graphs recursively by eliminating the failed components in the previous step. Decomposition continues recursively until all disjoint link sets

are identified. However, the number of sub-graphs will increase exponentially with the number of nodes in the graph, resulting in an exponential increase in computational cost in some cases.

Another method for analyzing the reliability of systems in complex configurations is through the use of Bayesian networks (BNs). One input required for the analyses is the set of minimum link sets (MLSs) or minimum cut sets (MCSs) of the system. Several efficient methods, e.g. EG-CUT algorithm for undirected graphs proposed by Shin and Koh (1998), have been developed to enumerate all MCSs or MLSs, which is an NP-hard problem (Suh and Chang 2000). By using a blocking mechanism repeatedly, MCSs can be generated at $O(en)$ per minimal cut set, where e is the number of edges and n the number of nodes in the graph. In Tien and Der Kiureghian (2016), BNs are used to probabilistically model system performance. Exact solutions for system reliability are achieved by performing inference calculations on the values in the conditional probability distributions defining the performance of each node in the BN. Computational limits in generating the BN for a general complex network, however, still exist (Tien and Der Kiureghian 2017), particularly for nodes in the BN with many parent nodes on which they depend.

MLSs and MCSs on their own can provide crude lower and upper bounds of system reliability. In Ebeling (2010), the reliability bounds of the system are determined by considering all MLSs to be in parallel (survival of any link set yields survival of the system) and all MCSs to be in series (failure of any cut set yields failure of the system). However, this method still relies on the generation of all MLSs and MCSs, which is NP-hard. In addition, the bounds provided by this method can be wide as it assumes that all MLSs and MCSs are independent of each other.

As an alternative to analytical solutions, simulation-based methods are widely used to assess the reliability of networks. Several sampling methods have been proposed to achieve improved efficiency in estimating low system failure probabilities, e.g., the random walk on graphs (Cheng et al. 2017) and the refined stratified sampling strategy (Shields et al. 2015). Bulteau and El Khadiri (1998) combine importance sampling and stratified Monte Carlo principles to generate nodal states. However, after a sample point is generated, it still needs to be tested against the MLSs or MCSs to determine the network outcome. Rejection sampling is used by Cheng et al. (2016) in parametric sensitivity analysis and approximation of probability of failure. Compared with direct Monte Carlo simulation and extended Monte Carlo simulation analysis, rejection sampling improves both accuracy and efficiency. However, this method requires finding a distribution from which to sample.

For flow network reliability measures, subset simulation also improves on accuracy and efficiency compared to basic Monte Carlo, e.g., the subset simulation-based network reliability analysis in Zuev et al. (2015). One of the main challenges, however, is calculating the indicator function, which defines the system state given states of the links. Although sampling size for subset simulation is small compared with traditional Monte Carlo, it is still expensive to evaluate the indicator function for each sample point. One of the advantages of our proposed method is the absence of an indicator function as the distribution of nodal states is determined by propagation across the network as described in the following section.

We propose a new analytical method, called PrPm, to obtain accurate and computationally tractable reliability assessments of general networks. The proposed method originates from the idea of belief propagation to perform inference in network graphs. Belief propagation is a message-passing algorithm that provides an exact solution for acyclic graphs. The reader is referred to Coughlan (2009) and Barber (2012) for more details on the method. In general, a message is calculated and passed to other nodes in the graph, where it is updated before continuing propagation. The message, which is a partial sum reusable for the marginalization, is obtained by calculating the marginal distribution of each unobserved node conditioned on any observed nodes. The message carried by a node is updated according to the message received from its direct neighbors. Based on the Hammersley-Clifford theorem, for nodes in the graph X, the joint distribution $p(X) = \frac{1}{Z}\prod_{c\in\xi}\Psi_{x_c}$, where Z is the normalization constant, ξ is the set of maximal cliques of the graph, and Ψ are the potential functions. The number of terms in the joint distribution $p(X)$ grows exponentially as the number of nodes in the network increases. The advantage of belief propagation is that marginal probabilities can be computed in a time that grows only linearly with the number of nodes in the system (Yedidia et al. 2003). However, for cases where the joint distribution $p(X)$ cannot be expressed explicitly, as for a general network, belief propagation loses its advantage.

3.3 Theoretical deduction

3.3.1 Probability propagation sequence

The objective of the proposed PrPm is to propagate the message throughout the entire network starting from the source node to the terminal node. To do this, the sequence of probability

propagation must be determined. The following terminology is used: if a node does not carry any message, it is labeled as a non-propagated node. Once a node receives a message from its neighbors, it is recognized as a propagated node. In each step of probability propagation, the message passes from the propagated node to its non-propagated direct neighbors. The sequence of nodes in receiving and passing the message is determined based on the three rules listed below:

1. Newly defined propagated nodes must be the direct neighbors of propagated nodes.

2. Newly defined propagated nodes should not separate any two non-propagated nodes, which guarantees that every node in the network is considered.

3. Newly defined propagated nodes should not connect with each other, which guarantees that every link in the network is considered.

An example of the propagation sequence determination is shown in Figure 3.3.1.1. The source and terminal nodes are marked as S and T, respectively. Note that S and T can occur anywhere in the network. The other nodes are numbered for clarity in the illustration. The method is applicable for both directed and undirected graphs as the derivation of the updating rules in the following section does not depend on the directivity of the links. In the case of directed links, such as one described by two unidirectional links where the reliability from a node i to node j differs from that from j to i, one would need only to specify two sets of link reliabilities for the two directions for R_1 and R_2 in the updating rules described later in Table 3.3.2.1 and Table

3.3.2.2. The method is also applicable for cyclic networks, an example of which is illustrated in Figure 3.3.1.1.

In Figure 3.3.1.1 and in the rest of chapter 3 we use three symbols to denote the different node types. The empty circle represents a non-propagated node that has not yet received any message. The solid circle represents a propagated node that will not be involved in any future massage passing. We name these as non-boundary nodes. The solid diamond represents a propagated node that will be involved in future propagation steps. We name these as boundary nodes.

The top left graph in Figure 3.3.1.1 shows the initial state. In it, the source node is the only node that carries a message and is labeled as a solid diamond. The remaining nodes are labeled as empty circles because they have not yet received any message. Following rule #1, S is ready to propagate its message to its direct neighbors, nodes 2, 8, 12, and 6. The next step of the propagation is shown in the second graph from top left. Note that if node S propagates to nodes 2, 8, 12, and 6 at the same time, node 1 will be separated from the terminal node, which violates rule #2. Therefore, the next nodes propagated are 2, 8, and 12. The next step of the propagation is shown in the third graph from top left. From nodes 2, S, and 12, if we pass the message to nodes 1, 6, and 11 in the same step, links $6-1$ and $6-11$ are excluded from the network, which violates rule #3. Therefore, the next nodes propagated are 1, 3, 9, 13, and 17. The propagation continues until reaching the final step of the propagation as shown in the bottom right graph. By receiving the message from boundary nodes 24 and 20, the reliability at the terminal node T is determined. All steps are shown in Figure 3.3.1.1. In some cases, including in the example network shown in Figure 3.3.1.1, there are multiple propagation sequences

satisfying the three rules above. For example, for the top right graph in Figure 3.3.1.1, you may choose $11 \rightarrow 19 \rightarrow 5$ or $5 \rightarrow 11 \rightarrow 19$, etc. As long as the sequence satisfies the propagation rules, it is acceptable. A rule of thumb is to prioritize propagating to nodes with one direct neighbor as this yields no approximation in the calculation.

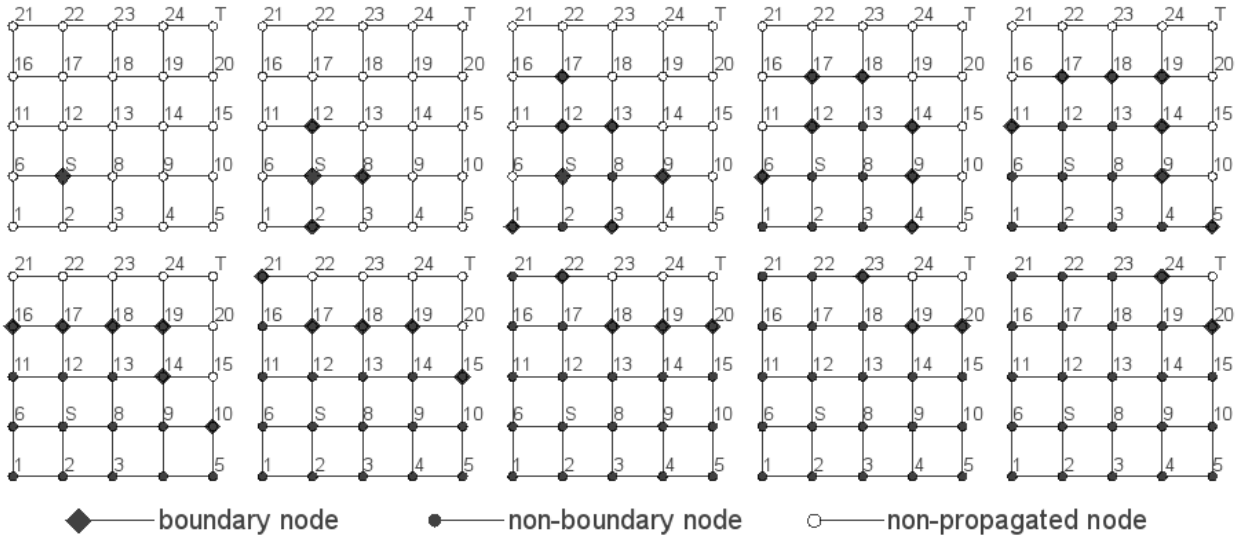

Figure 3.1.1.1 Propagation sequence illustration from source node *S* to terminal node *T*

3.3.2 Message passing and updating rules

We now discuss the message that is passed from node to node and how it is updated during propagation. We assume that each node receives messages from at most two direct neighbors. The situation where a node has more than two direct neighbors is addressed through a nodal expansion procedure presented in the following subsection. We also assume a binary network, i.e., one where nodes can be in one of two states such as 0 or 1 indicating failure or survival, respectively. For multi-state networks, the proposed PrPm is still workable if the multi-state network is converted into a binary state network. For example, we can classify a system of

multiple states as achieving or not achieving a certain level of service, or define survival as link flow capacity over a certain threshold and failure otherwise. Here, for message passing, two cases are considered: when a node receives a message from one direct neighbor as shown in Figure 3.3.2.1, or from two direct neighbors as shown in Figure 3.3.2.2. In these figures, we denote the node that receives a message as N, the direct neighbors that pass the message as A and B, and a general boundary node that is not a direct neighbor to N as C.

For the first case (Figure 3.3.2.1), node N receives a message from one direct neighbor A. The message is the joint distribution of the two nodes A and C from the previous propagation step. If it is the initial step, the message is the prior distribution of the source node. Reliability R_i denoted with a subscript indicates reliability of a link. The survival of node N is dependent on the survival of node A and the reliability of the link $A - N$ denoted R_1. The new discrete three-node joint distribution $p(A, C, N)$ is then derived using the updating rules shown in Table 3.3.2.1, where the distribution indicates the probability that each node is in one of two states, failure indicated 0 or success indicated 1. It is noted that no approximations are made in this calculation. Table 3.3.2.1 provides the updated probabilities that nodes A, C, and N are in each of the possible combinations of states 0 or 1. R denotes the reliability of node N, i.e., $R = P(N = 1)$, which is previously defined. Once the three-node joint distribution is obtained, we can easily define the new two-node joint distributions $p(A, N)$ and $p(C, N)$.

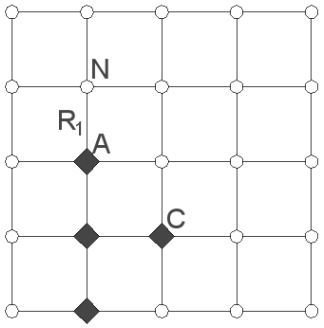

Figure 3.3.2.1 Message passing illustration from one direct neighbor

Table 3.3.2.1 Updating rules when receiving message from one direct neighbor

A	C	Pr		A	C	N	Updates
0	0	P_1		0	0	0	$P_1(1-R_1R)$
				0	0	1	P_1R_1R
0	1	P_2		0	1	0	$P_2(1-R_1R)$
			$\Longrightarrow$	0	1	1	P_2R_1R
1	0	P_3		1	0	0	$P_3(1-R_1R)$
				1	0	1	P_3R_1R
1	1	P_4		1	1	0	$P_4(1-R_1R)$
				1	1	1	P_4R_1R

For the second case (Figure 3.3.2.2), node N receives a message from two direct neighbors A and B. The message we need for the calculation is the joint distribution of the two nodes A and B and the marginal distribution of node C, which can be inferred from the two-node joint distribution

including node C. In updating the message, we assume that node C is separated from nodes A and B, which indicates that links $A-N$ and $B-N$ have no influence on node C. This underestimates reliability of node C. A detailed analysis of the error introduced by this assumption is provided later in section 3.3.7.

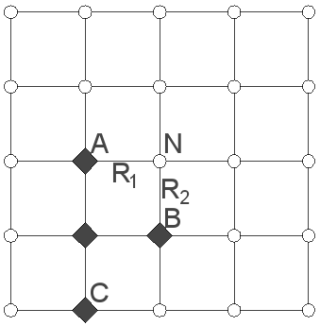

Figure 3.3.2.2 Message passing illustration from two direct neighbors

Table 3.3.2.2 Updating rules when receiving message from two direct neighbors

A	B	C	Pr		A	B	C	N	Updates
0	0	0	P_1		0	0	0	0	P_1
					0	0	0	1	0
0	0	1	P_2		0	0	1	0	P_2
					0	0	1	1	0
0	1	0	P_3	$\Rightarrow$	0	1	0	0	$P_3(1-R_2R)$
					0	1	0	1	$P_3(1-R_1)R_2R$
0	1	1	P_4		0	1	1	0	$P_4(1-R_2R)$
					0	1	1	1	$P_4(1-R_1)R_2R$
1	0	0	P_5		1	0	0	0	$P_5(1-R_1R)$

Table 3.3.2.2 continued

					1	0	0	1	$P_5RR_1(1-R_2)$
1	0	1	P_6		1	0	1	0	$P_6(1-R_1R)$
					1	0	1	1	$P_6(1-R_2)R_1R$
1	1	0	P_7		1	1	0	0	$P_7\{1-[1-(1-R_1)(1-R_2)]R\}$
					1	1	0	1	$P_3R_1R_2R+P_5RR_1R_2+P_7[1-(1-R_1)(1-R_2)]R$
1	1	1	P_8		1	1	1	0	$P_8\{1-[1-(1-R_1)(1-R_2)]R\}$
					1	1	1	1	$P_4R_1R_2R+P_6RR_1R_2+P_8[1-(1-R_1)(1-R_2)]R$

Table 3.3.2.2 shows the updating rules to build the four-node joint distribution $p(A, B, C, N)$ from the three-node joint distribution $p(A, B, C)$, where R, R_1, and R_2 indicate the reliabilities of node N, link $A - N$, and link $B - N$, respectively. The new joint distributions $p(A, N)$, $p(B, N)$, and $p(C, N)$ for future propagation steps can be defined accordingly. Based on the four-node joint distribution $p(A, B, C, N)$, $p(A, B)$, $p(A, C)$, and $p(B, C)$ are updated as well.

One important result from the updating rules given in Tables 3.3.2.1 and 3.3.2.2 is that we need the joint distributions of only two nodes rather than all nodes during the message-passing process. While this yields an approximated solution, PrPm reduces the computational cost from an exponential increase with the number of nodes in the network $O(2^n)$ to a quartic increase

$O(n^4)$. A detailed analysis of the computational complexity of the method is provided later in section 3.3.6.

3.3.3 Nodal expansion

The updating rules in Tables 3.3.2.1 and 3.3.2.2 are based on the assumption that every node receives a message from no more than two nodes. In a general network, however, it is possible that a node receives a message from a greater number of nodes. For example, as shown in Figure 3.3.3.1(a), a node i can have four or five direct neighbors. The updating in Tables 3.3.2.1 and 3.3.2.2 will not work for these configurations. However, we can expand the node i as shown in Figure 3.3.3.1(b). It is easy to prove that the configurations in Figure 3.3.3.1(a) are equivalent to the configurations in Figure 3.3.3.1(b), for which the previously derived updating rules are applicable. For example, for the four-neighbor case, instead of updating node i directly, we update the node sequentially $i_1 \rightarrow i_2 \rightarrow i_3 \rightarrow i_4$ as an alternative. Similarly, for the five-neighbor case, the updating rules are applicable if we update the node $i_1 \rightarrow i_2 \rightarrow i_3 \rightarrow i_4 \rightarrow i_5$ as shown on the right. In the proposed method, nodal expansion is performed before beginning the message passing. Without affecting the connectivity of original network, the additional links created by nodal expansion are set to be 100% reliable.

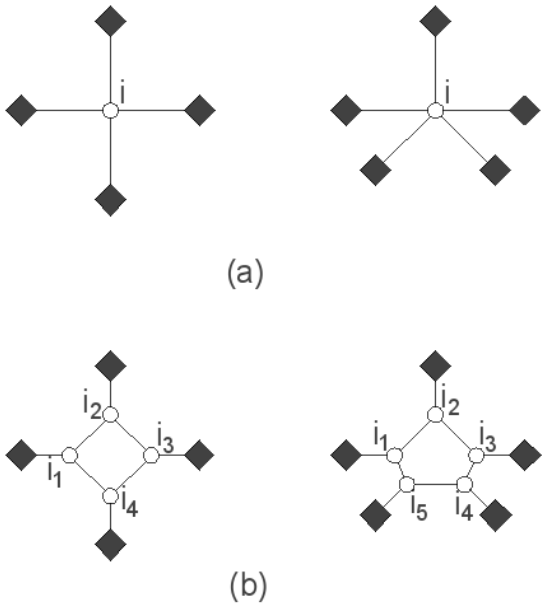

Figure 3.3.3.1 Nodal expansion illustration from multiple direct neighbors (a) to two direct neighbors (b)

3.3.4 Overall method

The full flowchart of the proposed PrPm is shown in Figure 3.3.4.1. First, we determine the propagation sequence based on the network configuration and propagation rules. This provides the sequence of steps in the probability propagation process for when and how each node receives the message from the other nodes. Then, we expand the nodes in the network as necessary to ensure that every node receives a message from at most two direct neighbors. Next is the message passing between nodes, where we define and update the message based on link and node reliabilities and the derived updating rules. After the message propagates to the terminal node, the approximated analytical solution of the network reliability is obtained.

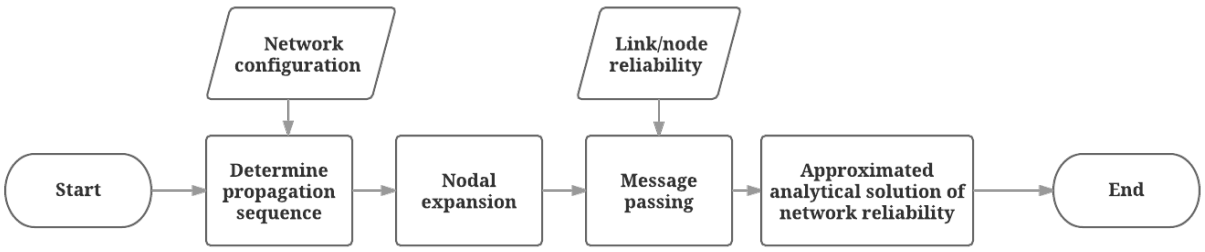

Figure 3.3.4.1 Flowchart of the proposed PrPm

3.3.5 Target networks

The proposed PrPm is applicable to directed, undirected, cyclic, and acyclic networks. In the case of directed networks, the method is able to analyze networks with bidirectional or unidirectional links as long as the individual link reliabilities are specified. PrPm works for both single-source-single-sink networks as shown in the first and third test applications, and multiple-sources-single-sink networks as shown in the second test application. For applications on networks with multiple sinks, for example as in Liu and Li (2009), evaluations can be done on each terminal node separately. Applicable networks of the proposed method should have independent or conditionally independent links or nodes as we do not consider the link conditional probabilities in calculating nodal joint distributions. However, the method can be applied to achieve significant computational savings for systems with dependent components by conditioning on common parents of the links or nodes.

3.3.6 Computational complexity analysis

The computational complexity of the method derives from the nodal expansion and updating rules. As we need to expand the node to ensure that every node receives information from at most two direct neighbors, for a network with n nodes, the newly defined propagated nodes connect to $O(n)$ neighbors with the maximum number being n for a fully connected network. Thus, there will be $O(n^2)$ nodes in total. According to the updating rules, for each newly defined propagated node, the computational cost for that node is $O(n^2)$, as the number of C nodes is $O(n^2)$ with the maximum number being $n^2 - 2$, excluding node A and node N as shown in Figure 3.3.2.2. Therefore, the total computational cost is the combined individual computational costs, $O(n^2)O(n^2) = O(n^4)$.

3.3.7 Error analysis

We now discuss the approximation error of the proposed method compared to the exact solution. The error in message passing arises from building the three-node joint distribution $p(A, B, C)$ from $p(A, B)$ and $p(C)$ as shown in Table 3.3.2.2 and the assumption that node C is separated from nodes A and B. The exact case and the extreme case for assessing the error are shown in Figure 3.3.7.1(a) and Figure 3.3.7.1(b), respectively. For the purposes of the illustration, the source node S is taken as the previously propagated node. In Figure 3.3.7.1(a), nodes A, B, and C are connected to S by three independent links with reliabilities R_1, R_2, and R_3. Based on the assumptions made in Table 3.3.2.2 that node C is independent of nodes A and B, PrPm will give us the exact joint probability in this scenario. However, Figure 3.3.7.1(b) shows the extreme case, i.e., the worst case in terms of error, where a common link with reliability R_4 is shared by paths from all nodes A, B, and C to S. The assumption in generating the joint distribution will not hold in this case because the reliability of node C is influenced by the states of nodes A and B.

For this case, the comparison between the exact distribution and the approximated distribution for PrPm is shown in Table 3.3.7.1.

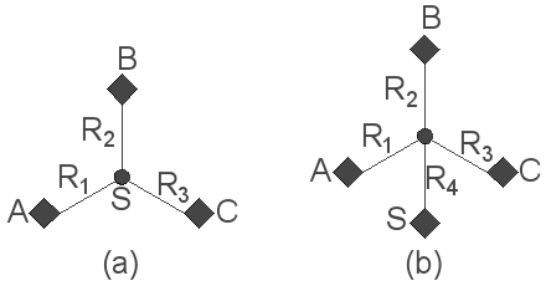

Figure 3.3.7.1 Illustration of the exact (a) and extreme (b) cases for error analysis

Table 3.3.7.1 Comparison between exact solution and distribution obtained from PrPm

A	B	C	Exact	PrPm	Difference	
0	0	0	$1+R_4(R_1R_2+R_1R_3+R_2R_3-R_1-R_2-R_3-R_1R_2R_3)$	$[1+R_4(R_1R_2-R_1-R_2)](1-R_3R_4)$	Δ_1	$(R_1+R_2-R_1R_2)R_3R_4(1-R_4)$
0	0	1	$(1-R_1)(1-R_2)R_3R_4$	$[1+R_4(R_1R_2-R_1-R_2)]R_3R_4$	Δ_2	$-(R_1+R_2-R_1R_2)R_3R_4(1-R_4)$
0	1	0	$(1-R_1)R_2(1-R_3)R_4$	$(1-R_1)R_2(1-R_3R_4)R_4$	Δ_3	$-(1-R_1)R_2R_3R_4(1-R_4)$
0	1	1	$(1-R_1)R_2R_3R_4$	$(1-R_1)R_2R_3R_4^2$	Δ_4	$(1-R_1)R_2R_3R_4(1-R_4)$
1	0	0	$R_1(1-R_2)(1-R_3)R_4$	$R_1(1-R_2)(1-R_3R_4)R_4$	Δ_5	$-R_1(1-R_2)R_3R_4(1-R_4)$
1	0	1	$R_1(1-R_2)R_3R_4$	$R_1(1-R_2)R_3R_4^2$	Δ_6	$R_1(1-R_2)R_3R_4(1-R_4)$
1	1	0	$R_1R_2(1-R_3)R_4$	$R_1R_2(1-R_3R_4)R_4$	Δ_7	$-R_1R_2R_3R_4(1-R_4)$
1	1	1	$R_1R_2R_3R_4$	$R_1R_2R_3R_4^2$	Δ_8	$R_1R_2R_3R_4(1-R_4)$

In Table 3.3.7.1, the rightmost column gives the difference between the exact and PrPm values for each element of the joint distribution $p(A, B, C)$, denoted Δ_i. Analyzing the expressions for Δ_i enables us to quantify and analyze the error in the approximation. Specifically, we see that $\Delta_1, \Delta_4, \Delta_6, \Delta_8$ are greater than 0, which means that we underestimate their probability shares in the proposed method; $\Delta_2, \Delta_3, \Delta_5, \Delta_7$ are less than 0, which means that we overestimate their probability shares. In addition, $\Delta_7 + \Delta_8 = 0$, indicating that the underestimation of the probability for the more likely-to-be-survived state ($A = 1, B = 1, C = 1$) is equal to the overestimation of the probability for the less likely-to-be-survived state ($A = 1, B = 1, C = 0$). Likewise, $\Delta_5 + \Delta_6 = 0$ and $\Delta_3 + \Delta_4 = 0$, indicating that the differences in their probability shares are reallocated from the more likely-to-be-survived states to the less-likely-to-be-survived states as well. This underestimates the reliability. The only contrary case is for Δ_1 and Δ_2, where $\Delta_1 + \Delta_2 = 0$. However, $\Delta_1 = \Delta_4 + \Delta_6 + \Delta_8$; therefore, the magnitude of the overestimation error equals the sum of the underestimation errors.

In addition, $P(R|A = 0, B = 0, C = 1) - P(R|A = 0, B = 0, C = 0) > P(R|A = 1, B = 1, C = 1) - P(R|A = 1, B = 1, C = 0)$, where $P(R|A, B, C)$ denotes the reliability of the network given the states of nodes A, B, and C. On the left-hand side of the inequality, terminal node T cannot be reached from nodes A and B; while, on the right-hand side of the inequality, terminal node T can be reached from nodes A and B. The inequality holds because paths from node C to terminal node T may share common links with paths from node A or B to T. For the same reason, $P(R|A = 0, B = 0, C = 1) - P(R|A = 0, B = 0, C = 0) > P(R|A = 0, B = 1, C = 1) - P(R|A = 0, B = 1, C = 0)$ and $P(R|A = 0, B = 0, C = 1) - P(R|A = 0, B = 0, C = 0) > P(R|A = 1, B = 0, C = 1) - P(R|A = 1, B = 0, C = 0)$. Since $\Delta_1 = \Delta_4 + \Delta_6 + \Delta_8$, it yields that

$[P(R|A=0,B=0,C=1)-P(R|A=0,B=0,C=0)]\Delta_1 > [P(R|A=1,B=1,C=1)-P(R|A=1,B=1,C=0)]\Delta_8 + [P(R|A=1,B=0,C=1)-P(R|A=1,B=0,C=0)]\Delta_6 + [P(R|A=0,B=1,C=1)-P(R|A=0,B=1,C=0)]\Delta_4$. This indicates the reallocation of the probability shares created by the proposed method overestimates the reliability of the network, resulting in the upper bound.

However, we also assume there is no connection between nodes A and B and node C. Under this assumption, C cannot be reached from A and B when links $A-N$ and $B-N$ are added as in Figure 3.3.2.2. This underestimates the connectivity of the network and tends the reliability toward the lower bound. These two effects, overestimation of the joint distribution $P(A,B,C)$ and underestimation of the reliability of node C, combine and cancel out the error to some extent. In practice, the actual error will fall between the errors given by the two extreme cases. Thus, the result obtained by the proposed method becomes a relatively accurate approximation to the exact solution as shown in the test applications.

A close look at the difference terms in Table 3.3.7.1 also reveals the performance of the method under high system reliability and low system reliability scenarios. All difference terms, Δ_1 to Δ_8, share a common factor $R_4(1-R_4)$. For both a highly reliable system and in a low reliability setting such as under a hazard, the term $R_4(1-R_4)$ tends to 0, reducing the error in these cases.

In addition, due to the source of the approximation error, the accuracy of the proposed PrPm increases as system failure probability decreases. This is in contrast to most sampling-based approaches. The sources of the error in PrPm compared to the exact solution are 1)

overestimation by the three-node joint distribution and 2) underestimation by assuming that C cannot be reached from A and B when links $A-N$ and $B-N$ are added as shown in Figure 3.3.7.1.

3.4 Test application – Seven-component network, Power distribution network and grid network

We now apply the proposed PrPm to three test applications and assess the performance in terms of accuracy and computational cost. All results are based on computations run in MATLAB_R2016b on a 16 GB RAM computer. All examples have exact solutions and computational costs for comparison with the proposed method. We demonstrate the procedure of calculation in detail in the first example, which is a simple single-source-single-sink network. A more complex, real-world, multiple-source-single-sink network is analyzed in the second example, including assessments of the system with increasing link reliabilities. A highly connected grid network is tested in the third example to assess the performance of the proposed method in terms of both accuracy and efficiency for systems of increasing size.

3.4.1 Seven-component network

First, we apply PrPm to an example from a previous study on network reliability (Tong and Tien 2017), which is shown in Figure 3.4.1. This network is chosen as it is irreducible to series and parallel components. It facilitates simple illustration of the method, and the exact solution can be obtained to compare accuracy with the result achieved by PrPm. For this example, the reliability of each link is assumed to be 0.9. It is noted that as PrPm calculates the network failure probability analytically, it is equally computationally efficient for varying link failure

probabilities of any value across the network, including for highly reliable networks with low probabilities of failure. For this example, nodes are considered to be perfectly reliable. In the figure, S and T represent the source and terminal nodes, respectively.

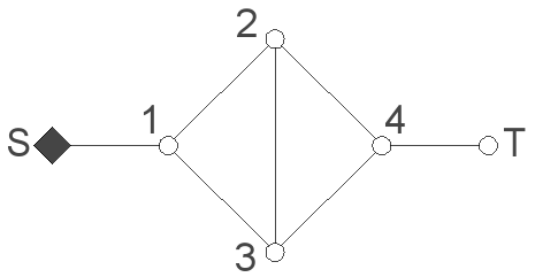

Figure 3.4.1 Example irreducible seven-component network

Following the overall process of the proposed PrPm shown in Figure 3.3.4.1, we first determine the propagation sequence, i.e., the order of nodes to receive messages by the propagation and updating rules. In this case, we pass messages following two possible sequences: $S \rightarrow 1 \rightarrow 2 \rightarrow 3 \rightarrow 4 \rightarrow T$ or $S \rightarrow 1 \rightarrow 3 \rightarrow 2 \rightarrow 4 \rightarrow T$. In either case, the sequence ensures every node is propagated before reaching the terminal node with both sequences working equally. For this network, no node requires nodal expansion.

We then pass the message through the network according to the probability distribution updating rules in Table 3.3.2.1 and Table 3.3.2.2. For illustration, we choose the first sequence above. We begin with node S. Initially, we have $P(S = 1) = 1$ and $P(S = 0) = 0$. The message is then updated to node 1, the direct neighbor of S. Let nodes $1, \dots, 4$ be denoted $N_1, \dots, N_4$. In this step, $P(N_1 = 1) = 0.9$ and $P(N_1 = 0) = 0.1$. Next, the message is updated to node 2 with $P(N_1 =$

$1, N_2 = 1) = 0.81$, $P(N_1 = 1, N_2 = 0) = 0.09$, $P(N_1 = 0, N_2 = 1) = 0$, and $P(N_1 = 0, N_2 = 0) = 0.1$. Node 3 is then updated with message $P(N_2 = 1, N_3 = 1) = 0.8748$, $P(N_2 = 1, N_3 = 0) = 0.0081$, $P(N_2 = 0, N_3 = 1) = 0.0081$, and $P(N_2 = 0, N_3 = 0) = 0.109$. Then, message is updated to node 4 with $P(N_4 = 1) = 0.8806$ and $P(N_4 = 0) = 0.1194$. Finally, we reach the terminal node with a message $P(T = 1) = 0.7926$ and $P(T = 0) = 0.2074$ to complete the reliability calculation.

In this case, as there is no C node during the propagation as shown in Figure 3.3.2.2, the result obtained by PrPm is the exact solution with no errors. As a comparison, we cite the results from Tong and Tien (2017), which provide an exact solution for the reliability of the network by a Bayesian network (BN) formulation. The comparison is given in Table 3.4.1.1. In this case, the reliability value computed using PrPm is exact. In terms of the computational cost, we see that computation time is reduced by more than two orders of magnitude or 500 times to arrive at the exact answer.

Table 3.4.1.1 Performance comparison for seven-component network between exact and PrPm solutions

	Reliability	Computation time (sec)
Exact solution (BN)	0.7926	35.00
PrPm solution	0.7926	0.06

3.4.2 Power distribution network

Next, we apply PrPm to a more complex system, which is the example four-substation power distribution network from Pacific Gas and Electric (Ostrom 2004) shown in Figure 3.4.2.1, also investigated in Der Kiureghian and Song (2008) and Tien (2017). The original system consists of 59 components, including circuit breakers, switches, and transformers. Each triplet configuration in the system of switch-breaker-switch can be easily represented as a single component. Therefore, 22 components are shown in Figure 3.4.2.1. For this example, all components are assumed to be independent and no nodal failure is considered. Previous studies assess network reliability based on varying component failure probabilities. Here, we convert to link failure probabilities. Compared with the original network, links $1-2$, $3-10$, $5-13$, $7-8$, $11-19$, $14-21$ and $16-18$ are assumed to be perfectly reliable as there are no additional elements on these links. All other links, which have circuit breakers, switches, and transformers located on them, have a probability of failure p_f. The network has multiple sources: nodes 1, 7, and 8; node T is the terminal node. Note that PrPm is able to accommodate the case of multiple source nodes across the network. As a reference, the method of total enumeration is used to obtain the exact solution. Results from Monte Carlo simulation are also provided for comparison. For this network, the existence of the C node as shown in Figure 3.3.2.2 during the message-passing process introduces errors into the propagation. Therefore, the results obtained by PrPm are an approximation in this case.

To investigate the accuracy and computational cost of PrPm over networks of varying reliabilities, we obtain results over a range of link failure probabilities. Table 3.4.2.1 provides the comparison among total enumeration to obtain the exact solution, Monte Carlo simulation, and PrPm. Results are given in terms of system reliability assessment and computation time (in

seconds) as p_f increases from 0.01 to 0.2. For Monte Carlo simulation, 10000 realizations are simulated for each p_f. From Table 3.4.2.1, PrPm outperforms Monte Carlo simulation in both accuracy and computation time. The percentage error relative to reliability for both PrPm and Monte Carlo decrease with smaller probabilities of failure. However, the accuracy relative to system failure probability decreases for Monte Carlo, as expected for simulation-based methods, while PrPm increases in accuracy as failure probabilities decrease as described in the error analysis section.

The network reliabilities obtained by PrPm and the exact solution are plotted in Figure 3.4.2.2 to show the trend in accuracy across link and system reliabilities. Over the investigated range of link failure probabilities, the maximum percentage error is 0.6357%, with decreasing error for systems of increasing reliability. As discussed in the error analysis section, errors should decrease for cases with low link reliabilities such as under hazard scenarios as well. As an additional comparison, if the link failure probability increases to 0.85, PrPm provides a solution with 0.0801% error, with the exact solution and PrPm indicating system failure probabilities of 0.9982 and 0.9974, respectively. In terms of computation, as PrPm provides an analytical solution, the burden of the method remains constant across system failure probabilities. Therefore, for all cases, PrPm increases the efficiency of obtaining the solution by more than three orders of magnitude and 1800 times, indicated as time ratio in Table 3.4.2.1, in terms of computation time.

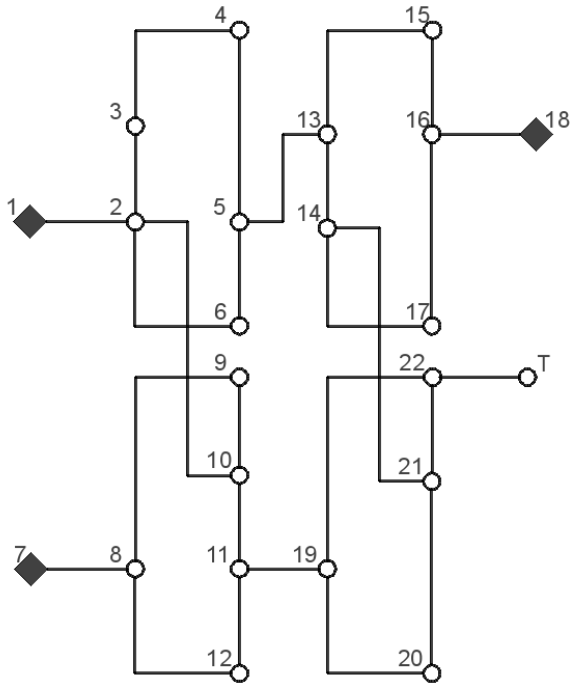

Figure 3.4.2.1 Power distribution network example

Table 3.4.2.1 Performance comparison for power distribution network among exact solution, PrPm, and Monte Carlo simulation varying p_f

	Exact solution		PrPm		Percentage error (%) relative to reliability	Percentage error (%) relative to failure probability	Time ratio
p_f	Reliability	Time (sec)	Reliability	Time (sec)			
0.0100	0.9899	113.63	0.9899	0.0629	0	0	1806.52
0.0500	0.9476		0.9474		0.0211	0.3817	
0.1000	0.8900		0.8888		0.1348	1.0909	
0.1500	0.8264		0.8233		0.3751	1.7857	
0.2000	0.7551		0.7503		0.6357	1.9600	

Table 3.4.2.1 continued

	Exact solution		Monte Carlo		Percentage error (%) relative to reliability	Percentage error (%) relative to failure probability	Time ratio
p_f	Reliability	Time (sec)	Reliability	Time (sec)			
0.0100	0.9899	113.63	0.9912	1.1013	0.1313	12.8713	103.18
0.0500	0.9476		0.9454		0.2322	4.1985	
0.1000	0.8900		0.8936		0.4045	3.2727	
0.1500	0.8264		0.8211		0.6413	3.0530	
0.2000	0.7551		0.7635		1.1124	3.4300	

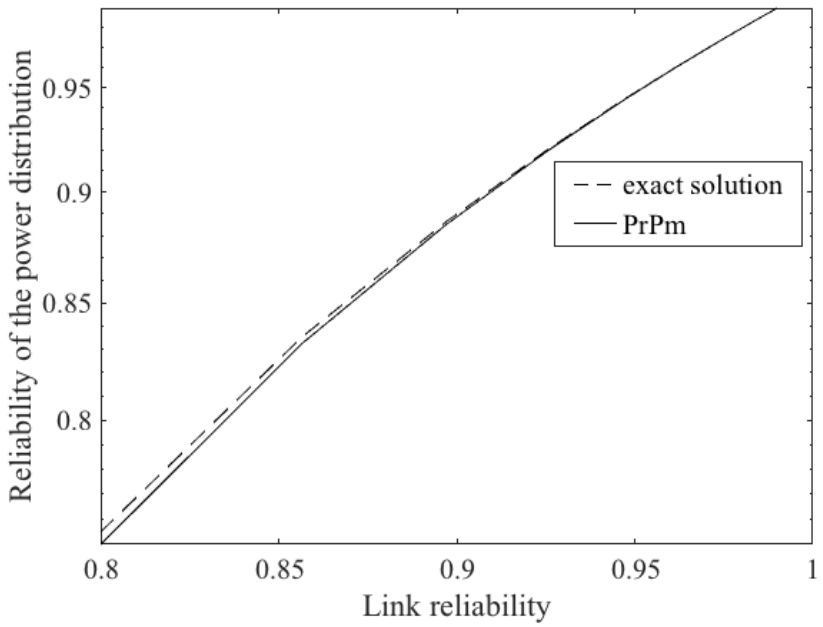

Figure 3.4.2.2 Exact solution compared to results by PrPm varying p_f

3.4.3 Grid network

Finally, we apply PrPm to analyze the reliability of a general grid network. First, we take a 5×5 grid as an example, with the source and terminal nodes at the corners of the grid as shown in Figure 3.4.3.1. Link failure probability p_f is assumed to be 0.1 and node reliability 1 across the network. We use the proposed PrPm and updating rules listed in Table 3.3.2.1 and Table 3.3.2.2 to obtain the approximated system reliability solution. For comparison, we use results from implementing the recursive decomposition algorithm (RDA). The RDA solution results in an upper bound and lower bound. By performing the decomposition recursively, the gap between the bounds is reduced to obtain a single system reliability value as shown in Table 3.4.3.1. Comparison between the performance of RDA and PrPm is listed in Table 3.4.3.1. With a sacrifice of 1.25% in accuracy, we reduce the computation time by more than four orders of magnitude and 35000 times for the 5×5 grid.

Figure 3.4.3.1 A 5×5 grid network

Table 3.4.3.1 Performance comparison for grid network between RDA and PrPm

RDA		PrPm		Percentage error (%)	Time ratio
Reliability	Time (sec)	Reliability	Time (sec)		
0.9755	1405.52	0.9877	0.0402	1.2564	35050.37

To explain the approximation error, we now provide further discussion on the error in propagating the two-node joint probability distribution compared to the full joint distribution in the context of analyzing the grid network reliability. As previously discussed, the error in the PrPm approximation comes from the C node in Figure 3.3.2.2 and using the distributions $p(A, B)$ and $p(C)$ to estimate $p(A, B, C)$ in Table 3.3.2.2. A more accurate result can be obtained by considering the joint distribution of all boundary nodes at each step. By making assumptions on the connectivity within the boundary nodes, i.e., based on whether or not the boundary nodes are connected with each other, we can find the upper bound and lower bound of the system reliability.

For example, suppose we have nodes $\{A, \dots, I\}$ configured as part of a network as shown in Figure 3.4.3.2. We use the joint distribution of all boundary nodes, $p(E, F, G, H, I)$, to update the joint distribution of the newly defined propagated nodes, $p(A, B, C, D)$. Let 0 again denote the failure of a node and 1 denote survival. When we update $p(A, B, C, D)$ from $P(E = 1, F = 1, G = 0, H = 1, I = 1)$, the upper bound of $p(A, B, C, D)$ can be found by assuming that nodes E and F, also nodes H and I, are connected; the lower bound of $p(A, B, C, D)$ can be found by assuming that there is no connection between nodes E and F, or between nodes H and I. A

similar strategy can be applied to the other node combinations of E, F, G, H, and I. By doing so, we can find the upper bound and lower bound of the system reliability at the terminal node.

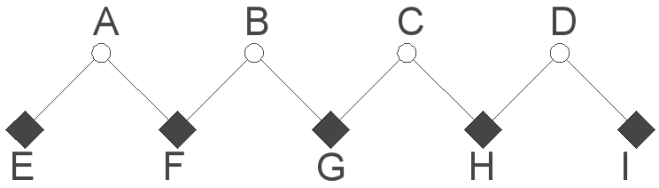

Figure 3.4.3.2 Illustration for considering the joint distribution of all boundary nodes

To quantify the effect of considering the joint distribution of all boundary nodes compared to the two-node distribution, we assess the accuracy and computation time to obtain the network reliability of grids of increasing size for the two cases. We take the corner-to-corner reliability of the grid network as the example, with link reliability of 0.9 and node reliability 1. Table 3.4.3.2 shows the results of the two-node joint distribution approximation compared to the upper and lower bounds considering the joint distribution of all boundary nodes as the size of the grid increases from 3×3 to 100×100. The full joint distribution calculation becomes intractable after a grid size of 12×12. In Table 3.4.3.2, the obtained bounds from the full joint distribution are guaranteed to include the exact solution. Computation times for both PrPm and the full distribution calculation are provided. Percentage error is calculated for the PrPm approximation result compared to the median of the bounds.

From Table 3.4.3.2, we see that when considering the full joint distribution, there is an exponential increase in computation time as the size of the grid increases. For a propagation step with n_b boundary nodes, calculating the joint distribution requires the storage and updating of

2^{n_b} elements, resulting in an exponentially increasing computational complexity with n at $O(2^{n_b})$. To improve the computational efficiency of reliability assessment of the network, the proposed method only considers the joint distribution of two nodes. With this, the accuracy of the result is slightly lowered by 1.24%, but the computational cost is reduced by several orders of magnitude, with computational savings increasing as the size of the network increases. With the consideration of the two-node joint distribution, the time complexity of computation for the proposed method is quartic at $O(n^4)$.

Table 3.4.3.2 Performance comparison for grid network between PrPm and considering the joint distribution of all boundary nodes

Grid size	PrPm		Full joint distribution			Percentage error (%)	Time ratio
	Reliability	Time (sec)	Reliability bounds		Time (sec)		
			Upper	Lower			
3×3	0.9833	0.1003	0.9725	0.9724	0.16	1.1157	1.60
4×4	0.9872	0.1092	0.9751	0.9750	0.26	1.2461	2.39
5×5	0.9877	0.1109	0.9756	0.9755	0.72	1.2455	6.49
6×6	0.9878	0.1162	0.9756	0.9756	2.83	1.2505	24.40
7×7	0.9878	0.1162	0.9757	0.9757	10.88	1.2401	93.79
8×8	0.9878	0.1162	0.9757	0.9757	42.98	1.2401	370.52
9×9	0.9878	0.1162	0.9757	0.9757	174.56	1.2401	1504.83
10×10	0.9878	0.1162	0.9757	0.9757	723.55	1.2401	6237.50

Table 3.4.3.2 continued

11×11	0.9878	0.1174	0.9757	0.9757	2986.20	1.2401	25523.08
12×12	0.9878	0.1287	0.9757	0.9757	12688.82	1.2401	98362.95
20×20	0.9878	0.2188	/	/	/	/	/
30×30	0.9878	0.6423	/	/	/	/	/
40×40	0.9878	1.7913	/	/	/	/	/
50×50	0.9878	4.6558	/	/	/	/	/
75×75	0.9878	46.1766	/	/	/	/	/
100×100	0.9878	196.0232	/	/	/	/	/

To further assess the performance of the method for general grid networks, Table 3.4.3.3 provides a comparison of the accuracy of PrPm compared to results from Monte Carlo simulation. The reader is referred to Dueñas-Osorio (2017) for details on the Monte Carlo simulation. For efficiency comparison, as the Monte Carlo simulations are tested on a different computer, they are not included here. For accuracy comparison, results in Table 3.4.3.3 are shown for systems with link failure probabilities of 10% and 1%. Although the average percentage error (0.24%) given by Monte Carlo simulation outperforms the average percentage error of 1.21% by PrPm for the case of link reliabilities of 0.9, Monte Carlo has a major limitation in that in the rare event condition, it can be computationally intractable to generate enough samples to calculate system reliabilities for low failure probability systems, e.g., systems with high link reliabilities. This is shown for networks with link reliabilities of 0.99. For the Monte Carlo simulation in this case, the result fails to converge under 7.8 hours of computation

for a 3×3 grid. In PrPm, computational efficiency is related only to the topology of the network and not influenced by the link reliability.

Table 3.4.3.3 Performance comparison for grid network between PrPm and Monte Carlo simulation

Grid size		PrPm		Monte Carlo simulation	
		Reliability	Percentage error (%)	Reliability	Percentage error (%)
$p_f = 0.1$	3×3	0.9833	1.1157	0.9729	0.0411
	4×4	0.9872	1.2461	0.9712	0.3897
	5×5	0.9877	1.2455	0.9787	0.3178
	10×10	0.9878	1.2401	0.9776	0.1947
$p_f = 0.01$	3×3	0.9998948	0.0102809	/	/
	4×4	0.9998979	0.0102050	/	/
	5×5	0.9998980	0.0102031	/	/
	10×10	0.9998980	0.0102031	/	/

3.5 Contributions

In this chapter, a new approximated analytical method, the probability propagation method (PrPm), is proposed to analyze the reliability of general networks. Compared to existing analytical algorithms such as RDA and inference in Bayesian networks, computational complexity with increasing nodes in the network n is reduced from an exponential increase $O(2^n)$ to a quartic increase $O(n^4)$. The method does not require the computationally intensive

enumeration of component states, MLSs, or MCSs to determine the system outcome. While the method results in an approximated value for network reliability with a small sacrifice in accuracy, compared with simulation-based methods, the proposed analytical PrPm solution does not require generating sample points or proving convergence. The source of the error in the proposed approximation is analyzed analytically, showing terms that both overestimate and underestimate the system reliability to effectively cancel out to obtain a solution. The performance of PrPm is investigated using three example networks. In the first example, PrPm results in the exact solution as all boundary nodes are direct neighbors at each step. In the second example, PrPm is shown to work for a network with multiple sources. Computational time is reduced by more than 1800 times with a maximum error in the reliability result of 0.64% compared to the exact solution. In the last example, the results show that the computation time does not exponentially increase with system size as with other methods, and the error is stable. Many sampling-based approaches are limited by computational tractability to analyze rare events. For PrPm, as the method calculates the network reliability analytically, it is equally computationally efficient across reliability values. Throughout, the proposed PrPm achieves accurate estimates of network reliability with orders of magnitude savings in computation time. This enables accurate and computationally tractable reliability assessments of larger, complex networks.

Chapter 4 Connectivity – directed Probability Propagation method (dPrPm)

The rest of this chapter is organized as follows. In the beginning, we provide background on analytical and sampling-based system reliability assessment methods, and the origin of the idea of dPrPm. The proposed dPrPm is detailed in section 4.3, including descriptions of propagation sequences and probability updating rules. Proofs are given to guarantee the upper and lower bounds for acyclic directed networks. This chapter then applies dPrPm to three examples: a directed grid network, power distribution network, and gas pipeline network. Results are shown to compare performance between the proposed method and existing methods in terms of accuracy and computation time for network reliability analysis. The proposed method (dPrPm) has been published by Tong and Tien (2019).

4.1 Introduction – reliability analysis on acyclic directed networks

Many civil infrastructure systems, e.g. power distribution and gas pipeline networks, are characterized by flow across the system. Considering system components as nodes and the connections between them as links, these infrastructures can be modeled as acyclic directed networks, with resources directed through the network from supplies to distribution points. Policies regarding inspection, maintenance, and replacement of components in the system rely heavily on reliability assessment of node accessibility, measuring the performance and reliability of the network.

Methods for reliability analysis include analytical and simulation-based methods. While analytical approaches usually produce an exact result, enumeration of minimum link sets

(MLSs), minimum cut sets (MCSs), or possible component states are often necessary to conduct an analysis. Increasing the number of nodes in the network typically yields an exponential increase in computational cost. For simulation-based methods, one of the challenges is to capture rare events. For networks that are highly reliable or highly unreliable, failure to capture sample points of rare events will result in high errors. Both the efficiency and accuracy of simulation-based methods vary on the reliability of links.

To address these challenges in system reliability assessment, a new analytical method, called the Probability Propagation method (PrPm), was proposed (Tong and Tien 2018). The method provides an estimate of network reliability for general complex networks. However, there are no guarantees on the accuracy of the estimate. Rather than providing an approximation, here, an advancement on PrPm called the directed Probability Propagation method (dPrPm) is proposed to provide reliability assessment of directed acyclic networks with guaranteed upper and lower bounds. This requires new definitions of the propagation sequence and probability updating rules for message propagation. The outcome of dPrPm is the upper and lower bounds of all sink node reliabilities, given with 100% confidence level. The computational cost is independent of link reliabilities. dPrPm reduces the cost from a typical exponential increase with number of a nodes in a network to a polynomial increase.

4.2 Literature review – reliability analysis methods

This section briefly introduces some existing methods for system reliability analysis. For more in-depth study, readers are referred to texts such as Birolini (2004). Systems that can be reduced to fundamental parallel and series configurations can be readily analyzed. Analysis becomes

complicated under complex configurations, e.g., general infrastructure networks represented as acyclic directed networks. Among analytical methods, one brute force approach is to enumerate all possible combinations of component states and determine their outcomes accordingly. As the computational cost increases exponentially with the size of the system, total enumeration becomes computationally intractable for large networks.

To address this issue, efforts have been made to improve the computational efficiency and storage requirements of enumeration-based methods. In Tien and Der Kiureghian (2016) and Tong and Tien (2017), compression and inference algorithms have been proposed to facilitate inference in binary and multi-state Bayesian network representations of infrastructure systems. However, computational limits on system size still exist, particularly for networks with a large number of parent nodes (Tien and Der Kiureghian 2017). In Ebeling (2010), the reliability bounds are determined by taking all MLSs in parallel (survival of any link set yields survival of the system) and all MCSs in series (failure of any cut set yields failure of the system). The gap between the upper and lower bounds can be wide, however, because of the independency assumption among all MLSs and MCSs. In addition, the use of both Bayesian networks and the bound-finding method require identification of the MLSs or MCSs of a system. While methods have been proposed to do this efficiently, e.g., the edge cut algorithm, also known as EG-CUT algorithm, to generate MCSs in Shin and Koh (1998) and a recursive depth-first search MLS identification algorithm in Applegate and Tien (2018), this is a NP-hard problem (Suh and Chang 2000) for general networks.

Recursive decomposition algorithm (RDA), e.g. Dotson and Gobien (1979), provides an alternative analytical approach. In Lim and Song (2012), selective RDA is proposed to improve the efficiency of original RDA by identifying the most reliable paths. However, the computational cost may still follow an exponential increase, particularly when the most reliable paths are not significantly more dominant than others. In Kim and Kang (2013), application of RDA is extended to multiple-source-multiple-sink situations. Multiple sinks are connected to one aggregated sink node, however, which loses the reliabilities of individual sink nodes. Similar limitations are found in Liu and Li (2009). In essence, the focus is still on a one-sink node network. In addition, the number of subgraphs created by eliminating the failed components still increases exponentially with the system size.

For simulation-based methods, it is often a challenge to efficiently capture the occurrence of rare events. To address this, several sampling methods, e.g., refined stratified sampling strategy (Shields et al. 2015), rejection sampling from predetermined distribution (Cheng et al. 2016), and random walk on graphs (Cheng et al. 2017) have been proposed. Although the sampling size and efficiency have been improved, challenges still exist in determining the system outcome. In comparison, the efficiency and accuracy of PrPm (Tong and Tien 2018) and dPrPm as described in chapter 4 are not limited by the ability to assess low probability events. In Bulteau and El Khadiri (1998) and Zuev el al. (2015), the result of a sampling point depends on an indicator function, e.g., by comparison with MLSs or MCSs, which can be expensive to evaluate. One advantage of the proposed directed probability propagation method (dPrPm) is the absence of an indicator function or need to identify the MLSs or MCSs of a system.

The idea of PrPm and dPrPm originates from belief propagation. Details on belief propagation for network graphs are provided in Coughlan (2009) and Barber (2012). In chapter 4, the message that is passed from node to node refers to the marginal and pairwise node reliabilities. To increase computational efficiency, we do not consider the full nodal joint distributions. Accuracy is guaranteed through definition of the propagation sequence to create intermediate structures for final inference and the corresponding probability updating rules.

PrPm was first introduced in Tong and Tien (2018) and described in chapter 3. The method described in this chapter advances upon that work in three main ways: first, while PrPm could be used for single-source-single-sink networks and multiple-sources-single-sink networks, the case of multiple-sources-multiple-sinks networks was not addressed. In an infrastructure system, the multiple sinks scenario is particularly relevant to calculate probabilities of providing the infrastructure resource at multiple end-point distribution nodes. The directed PrPm, i.e., dPrPm, proposed in this chapter addresses the multiple sinks case and provides reliabilities at all sink nodes in the network. Second, while PrPm provided approximations of the network reliability that were shown to be accurate compared to the exact solution, there were no guarantees on the accuracy of the estimates nor were there bounds that could be placed on the results. Results were empirically shown to be accurate rather than being able to provide proofs of their accuracy. dPrPm provides 100% confidence bounds on the reliabilities at sink nodes and proofs are provided guaranteeing their accuracy. Third, dPrPm focuses on reliability assessment of acyclic directed networks as applicable to infrastructure systems. In these systems, resources are distributed through a network from supply or generation nodes to distribution nodes without

cycles in the network. The newly described propagation sequences and probability updating rules reflect this case.

4.3 Theoretical deduction

Notations

m: Number of links in the network

n: Number of nodes in the network

α: Lower case Greek letter refers to a node in the network

A: Upper case letter refers to node type

$l_{\alpha\to\beta}$: Link connecting node α to node β

$R_{\alpha\to\beta}$: Reliability of the link connecting node α to node β

$pa_{\alpha\to\beta}$: Path from node α to node β

$Pr(\alpha)$: Probability that node α is reachable

$Pr(\alpha,\beta)$: Probability that both node α and node β are reachable

$Pr(\alpha,\beta=0)$: Probability that node α is reachable but node β is not reachable

$Pr(\alpha|\beta)$: Probability that node α is reachable given node β is reachable

$Pr(\underline{\alpha})$: Lower bound of $Pr\ (\alpha)$

$Pr(\bar{\alpha})$: Upper bound of $Pr\ (\alpha)$

$\overline{Pr}$: Probability after updating

4.3.1 Networks of interest

Due to the characteristics of flows in infrastructure networks, we consider the links connecting the nodes in the system as directed. The reliability of a sink node in the system is defined as its accessibility from a source node. The links in the system are assumed to be independent or conditionally independent of each other. In our proposed method, dPrPm, there is no limitation on the number of source nodes and sink nodes. An example 16-node network is shown in Figure 4.3.1.1, where solid black circles are source nodes and all empty circles are sink nodes. Arrows on links indicate directionality of flow. In the following sections, we will use the network shown in Figure 4.3.1.1 as an example to illustrate the procedure of dPrPm.

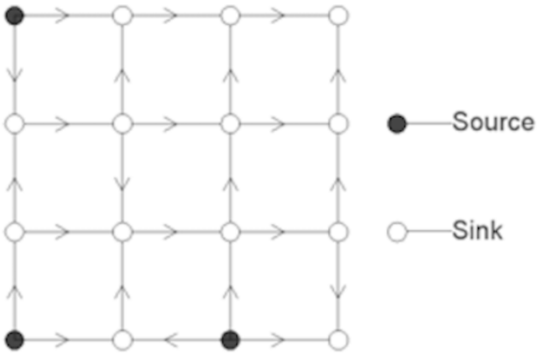

Figure 4.3.1.1 Example network of interest

4.3.2 Overall method

In belief propagation, messages are passed from node to node in the network to perform inference. An example is shown in Figure 4.3.2.1, where node A receives the message from node B and node C. It updates the message and then passes the message to node D. In the case where the structure is a tree, belief propagation yields an exact marginal distribution.

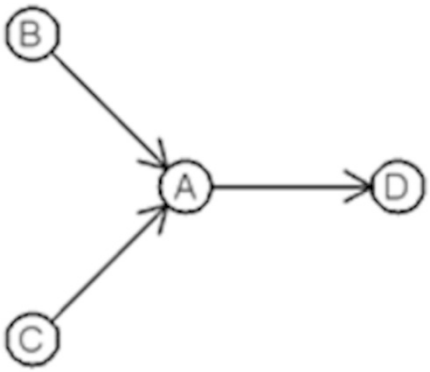

Figure 4.3.2.1 Illustration of belief propagation

In dPrPm, we start the message propagation from source nodes in the network. The message being passed here refers to the marginal node reliability $Pr(\alpha)$ and pairwise node reliability $Pr(\alpha, \beta)$. To begin, we remove all links in the network and keep only the source nodes as shown in Figure 4.3.2.2(a). Then, we add in link one at a time, e.g., as in Figure 4.3.2.2(b) and Figure 4.3.2.2(c), to restore the original connectivity of the network. The configurations in Figure 4.3.2.2(b) and Figure 4.3.2.2(c) are referred to as intermediate structures in the remainder of this chapter. Every time a link is added, we update the message $Pr(\alpha)$ and $Pr(\alpha, \beta)$ for all nodes. In the end, once all links are added, as shown in Figure 4.3.2.2(d), the reliability of each sink node is obtained.

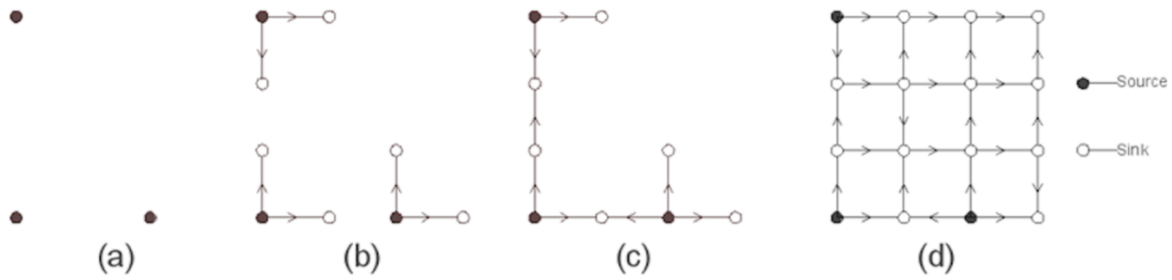

Figure 4.3.2.2 Constructing intermediate structures for directed probability propagation method (dPrPm) to obtain reliabilities at all sink nodes

4.3.3 Heuristic propagation sequences for acyclic directed networks

The message propagates through the network through a link-adding sequence. The defined sequence will affect the accuracy of the result. When a link is added to create a new intermediate structure, e.g., adding the dashed link $l_{\alpha\to\beta_1}$ to connect nodes α and β_1 from Figure 4.3.3.1(a) to Figure 4.3.3.1 (b), the reliabilities of multiple sink nodes are influenced. For example, in the case shown in Figure 4.3.3.1, after link $l_{\alpha\to\beta_1}$ is added, additional paths to nodes β_1, β_2, and β_3 are created. Thus, the message relating to these three nodes (β_1, β_2, and β_3) needs updating. However, as we will see in the next few sections, we cannot obtain exact values for these updating terms because the message inherited from the previous step only includes the marginal and pairwise node reliabilities. Approximations are necessary for estimating a three-node joint distribution.

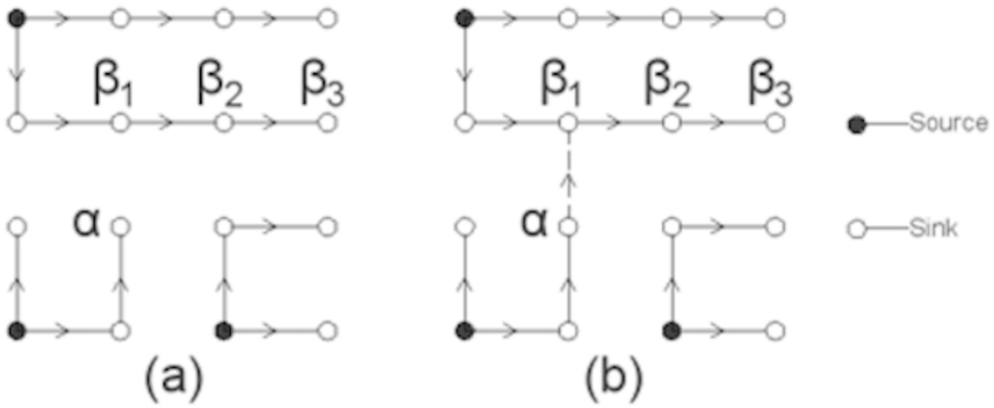

Figure 4.3.3.1 Adding link $l_{\alpha\to\beta_1}$ to form a new intermediate structure

To yield a more accurate solution, the objective is to make fewer approximations during message propagation. A heuristic for making fewer approximations is to limit the number of nodes influenced when links are added. As an example, in Figure 4.3.3.2, node β is the only node influenced after adding link $l_{\alpha\to\beta}$. Thus, we need only to update the message relating to node β.

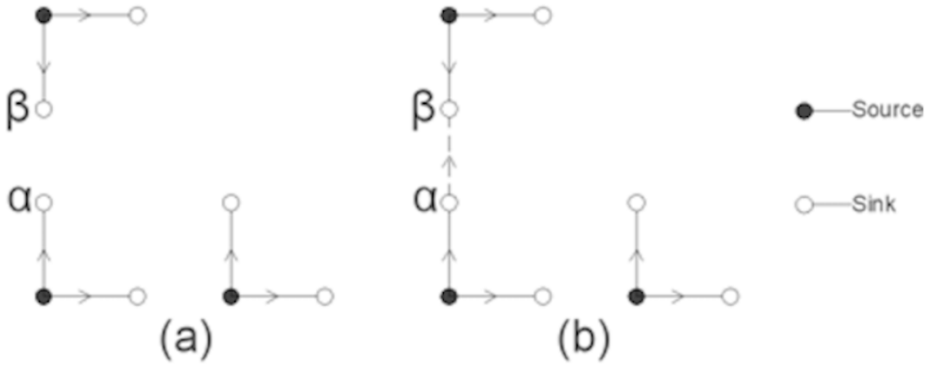

Figure 4.3.3.2 Adding link heuristically to reduce number of nodes influenced

For an acyclic directed network, we can find certain link-adding sequences that ensure that only one node is influenced every time a link is added. To illustrate this, we classify all nodes into one of three types in each intermediate step:

Type A node: In the intermediate structure, all links connected to the node have been added.
Type B node: In the intermediate structure, all incoming links to the node have been added.
Type C node: All other nodes.

Figure 4.3.3.3(a) gives the original connections in the network; Figure 4.3.3.3 (b) and Figure 4.3.3.3 (c) are intermediate structures. Node types are marked next to each node. At the start, as indicated in Figure 4.3.3.3 (b), no links are directed towards the source nodes, i.e., they have no incoming links that need to be added. The source nodes have outgoing links that have not yet been added. Therefore, all source nodes belong to type B. In Figure 4.3.3.3 (c), all links connected to type A nodes have been added. All incoming links to type B nodes have been added. The remaining nodes are type C.

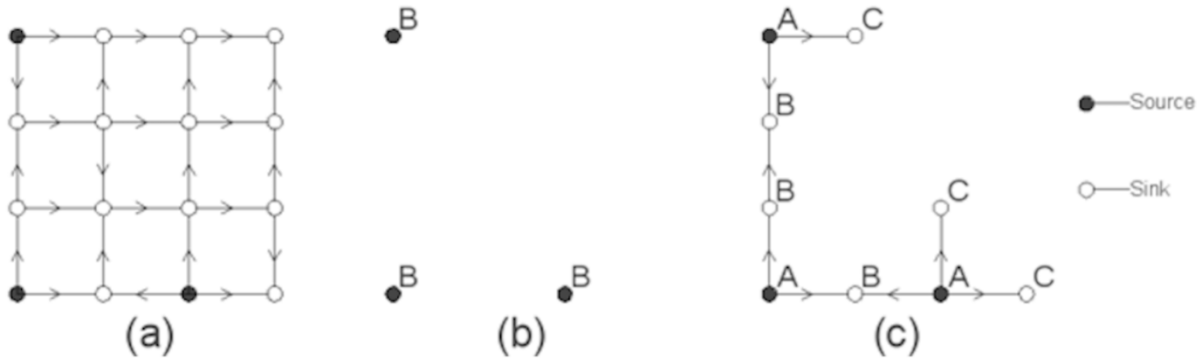

Figure 4.3.3.3 Example node classifications in intermediate structures (b) and (c)

We now provide two proofs showing the ability to define propagation sequences for acyclic directed networks such that only one node is influenced at each step of the link-adding sequence. This is done by always adding links originating from type B nodes.

Theorem 1: All intermediate structures have at least one type B node.

Proof: Suppose for a given intermediate structure, all nodes belong to either type A or type C. Let N be the number of type C nodes, as shown in Figure 4.3.3.4 (a), with the type C nodes in the structure indicated by the set $\{\alpha_1, \dots, \alpha_N\}$. $N \geq 1$, otherwise all nodes are type A and all links have been added. Based on the definition of a type C node, there is a path $pa_{\beta_1 \to \alpha_1}$ directed to α_1. Since the network is acyclic, β_1 cannot be α_1 and we set it as α_2. Similarly, for node α_2, there is a path $pa_{\beta_2 \to \alpha_2}$ directed to α_2. For the acyclic network, β_2 cannot be α_1 or α_2 and we set it as α_3. Continuing the deduction to α_N, there is a path $pa_{\beta_N \to \alpha_N}$ directed to α_N. As $\beta_N \in \{\alpha_1, \dots, \alpha_N\}$, it will create a loop, contradicting the acyclic assumption. Thus, at least one of the nodes in set $\{\alpha_1, \dots, \alpha_N\}$ belongs to type B. ■

Theorem 2: If we prioritize the links that originate from type B nodes in adding sequences, only one node is influenced every time a link is added to the intermediate structure.

Proof: Suppose a link $l_{\alpha \to \beta}$ is added, e.g., as shown in Figure 4.3.3.4(b), then node α is type B, and node β must be type C. The latter is because if we are adding a link into β, then clearly not all links into β have been added, and β is by definition a type C node. As we do not add links originating from type C nodes (in the example, node β), then β has no outgoing links and no

node can be reached from node β in the current intermediate structure. Thus, node β is the only node influenced. ■

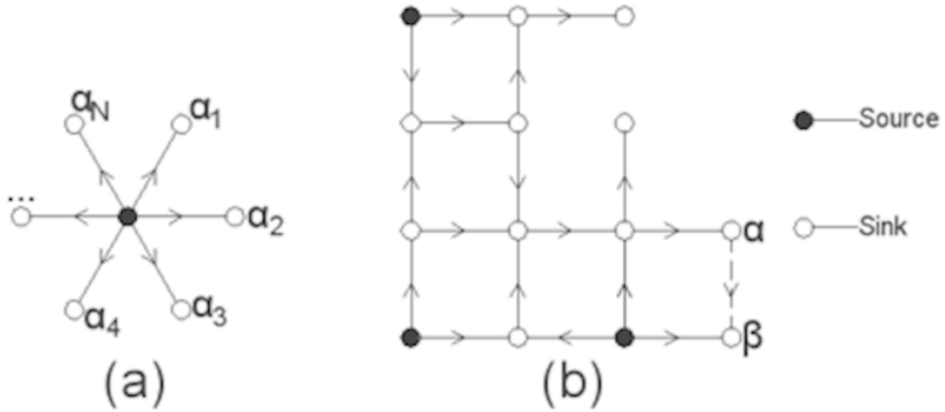

Figure 4.3.3.4 Intermediate structures for acyclic directed networks proofs

In sum, for acyclic directed networks, there always exists a link-adding sequence for which only one node is influenced for each added link, and hence makes the fewest approximations. Note that there can be multiple link-adding sequences satisfying the requirements above.

4.3.4 Message propagation: probability updating rules

As we add links and create intermediate structures, updates on the message should be made simultaneously. In dPrPm, the updating rules are discussed under two different scenarios as shown in Figure 4.3.4.1. In the figure, link $l_{\alpha\to\beta}$ is added into the intermediate structure and node γ is a random node other than node α or β. In Figure 4.3.4.1(a), β is not in the previous intermediate structure; in Figure 4.3.4. (b), it is. In both scenarios, it is easy to see that we need

only to update messages relating to node β. For notation, we use $\overline{Pr}$ to refer to terms after updating and Pr to terms before updating.

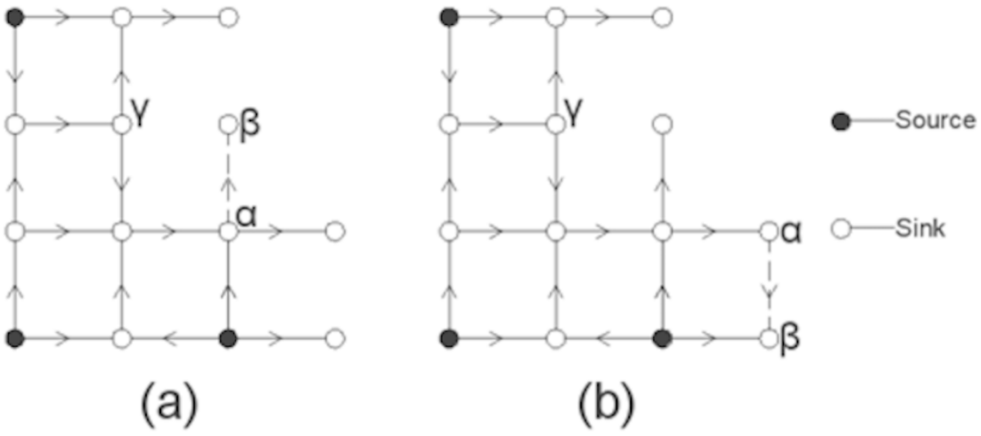

Figure 4.3.4.1 Two updating scenarios for added link $l_{\alpha\to\beta}$

In the first scenario, as shown in Figure 4.3.4.1(a), as node β is not in the previous intermediate structure, no approximation is needed. The updating rules are:

$$\overline{Pr(\beta)} = Pr(\alpha)\, R_{\alpha\to\beta} \qquad \ldots\ldots(1)$$

$$\overline{Pr(\alpha,\beta)} = Pr(\alpha)\, R_{\alpha\to\beta} \qquad \ldots\ldots(2)$$

$$\overline{Pr(\gamma,\beta)} = Pr(\alpha,\gamma)\, R_{\alpha\to\beta} \qquad \ldots\ldots(3)$$

In the second scenario, as shown in Figure 4.3.4.1(b), node β is already part of the previous intermediate structure. To calculate the updating rules, first, we construct the three-node joint distribution as shown in Table 4.3.4.1, where the probability shares for each combination of nodal states are given. We use X to represent the probability share where all three nodes, α, β,

and γ, are reachable, i.e., $Pr(\alpha, \beta, \gamma)$. For the remaining terms, they are inferred from the marginal and pairwise node distributions in the message. In Table 4.3.4.1, for the states of α, β, and γ, we use 1 to denote that the node is reachable and 0 otherwise.

Table 4.3.4.1. Three-node joint distribution

α	β	γ	Probability shares
0	0	0	X_1
0	0	1	X_2
0	1	0	X_3
0	1	1	X_4
1	0	0	X_5
1	0	1	X_6
1	1	0	X_7
1	1	1	X

$$X_1 = 1 - Pr(\alpha) - Pr(\beta) - Pr(\gamma) + Pr(\alpha, \gamma) + Pr(\alpha, \beta) + Pr(\beta, \gamma) - X$$

$$X_2 = Pr(\gamma) - Pr(\alpha, \gamma) - Pr(\beta, \gamma) + X$$

$$X_3 = Pr(\beta) - Pr(\beta, \gamma) - Pr(\alpha, \beta) + X$$

$$X_4 = \Pr(\beta, \gamma) - X$$

$$X_5 = Pr(\alpha) - Pr(\alpha, \gamma) - Pr(\alpha, \beta) + X$$

$$X_6 = Pr(\alpha, \gamma) - X$$

$$X_7 = Pr(\alpha, \beta) - X$$

$X = Pr(\alpha, \beta, \gamma)$

The updating rules for the scenario shown in Figure 4.3.4.1(b), including the unknown parameter X:

$$\overline{Pr(\beta)} = Pr(\beta) + (Pr(\alpha) - Pr(\alpha, \beta))R_{\alpha \to \beta} \quad \ldots\ldots(4)$$

$$\overline{Pr(\alpha, \beta)} = Pr(\alpha, \beta) + (Pr(\alpha) - Pr(\alpha, \beta))R_{\alpha \to \beta} \quad \ldots\ldots(5)$$

$$\overline{Pr(\beta, \gamma)} = Pr(\beta, \gamma) + (Pr(\alpha, \gamma) - X)R_{\alpha \to \beta} \quad \ldots\ldots(6)$$

To calculate the exact value of X, we would need the full three-node joint distribution. For computational and memory storage efficiency, we only focus on the marginal and pairwise node reliabilities and do not carry three-node joint distributions in the message. However, we are able to bound the value of X based on the following derivations from probability properties.

First, all the terms in the rightmost column in Table 4.3.4.1 should be greater than or equal to zero. Thus, let

$$X_1 = max\{Pr(\alpha, \gamma) + Pr(\beta, \gamma) - Pr(\gamma), Pr(\alpha, \beta) + Pr\ (\beta, \gamma) - Pr(\beta), Pr(\alpha, \gamma) + Pr(\alpha, \beta) - Pr(\alpha)\} \quad \ldots\ldots(7)$$

$$X_2 = min\ \{Pr(\alpha, \beta)\ , Pr(\beta, \gamma)\ , Pr(\alpha, \beta)\ , 1 - Pr(\alpha) - Pr(\beta) - Pr(\gamma) + Pr(\alpha, \beta) + Pr(\alpha, \gamma) + Pr\ (\beta, \gamma)\} \quad \ldots\ldots(8)$$

Then, the bounds of X are:

$$X_1 \leq X \leq X_2 \quad \ldots\ldots(9)$$

Second, by the definition of conditional probability and the coherency property of networks,

$X = Pr(\alpha, \beta, \gamma) = Pr(\alpha, \beta|\gamma)\, Pr(\gamma) \geq Pr(\alpha, \beta) Pr(\gamma)$ ……(10)

Let $X_3 = max\{Pr(\alpha, \beta)\, Pr(\gamma)\,, Pr(\alpha, \gamma)\, Pr(\beta)\,, Pr(\beta, \gamma)\, P\, r(\alpha)\} \leq X$ ……(11)

Similarly, $Pr(\alpha = 0, \beta = 0, \gamma) = Pr(\alpha = 0, \beta = 0|\gamma)\, Pr(\gamma) \leq Pr(\alpha = 0, \beta = 0)\, Pr(\gamma)$

Thus, $Pr(\alpha = 0, \beta = 0, \gamma) = Pr(\gamma) - Pr(\alpha, \gamma) - Pr(\beta, \gamma) + X \leq Pr(\alpha = 0, \beta = 0)\, Pr(\gamma)$

Let, $t_1 = Pr(\alpha = 0, \beta = 0)\, Pr(\gamma) - Pr(\gamma) + Pr(\alpha, \gamma) + Pr(\beta, \gamma)$,

$t_2 = Pr(\alpha = 0, \gamma = 0)\, Pr(\beta) - Pr(\beta) + Pr(\alpha, \beta) + Pr(\beta, \gamma)$,

$t_3 = Pr(\beta = 0, \gamma = 0)\, Pr(\alpha) - Pr(\alpha) + Pr(\alpha, \beta) + Pr(\alpha, \gamma)$

Then, $X \leq X_4 = min\{t_1, t_2, t_3\}$ ……(12)

Combining the inequality constraints in equations (9), (11), and (12),

$max\{X_1, X_3\} \leq X \leq min\{X_2, X_4\}$ ……(13)

Thus, we obtain the upper bound and lower bound of X. These derivations enable us to guarantee bounds on the node reliability values calculated through dPrPm as shown in the following:

Theorem 3: If we assign the lower bound value to X every time we update the message, we will obtain the lower bound values of marginal node reliabilities. If we assign the upper bound value to X every time we update the message, we will obtain the upper bound values of marginal node reliabilities.

Proof: We prove this theorem separately for the two updating scenarios shown in Figure 4.3.4.1. For the scenario in Figure 4.3.4.1(a), no approximation is made. As a result, the updated terms from equation (1) to (3) are not influenced by the estimation on X.

For the scenario in Figure 4.3.4.1(b), if we take the lower bound value of X, then based on equation (6), $\overline{\Pr(\beta,\gamma)}$ is overestimated. In the future updating steps, when link $l_{\gamma\to\beta}$ is added, equation (4) becomes $\overline{Pr(\beta)} = Pr(\beta) + (Pr(\gamma) - Pr(\beta,\gamma))R_{\gamma\to\beta}$. $\overline{Pr(\beta)}$ is then underestimated at the lower bound because of the overestimation on $Pr(\beta,\gamma)$. The same logic applies to taking the upper bound value of X, which yields an overestimation and upper bound on marginal node reliabilities.

■

The bounds can be further refined considering the possible link-adding sequences satisfying the requirement that only one node be influenced at each link-adding step. Suppose there are N link-adding sequences available for a general acyclic directed network. For the i^{th} sequence, we bound the marginal node reliability of node α as: $Pr(\underline{\alpha})_i \le Pr(\alpha) \le Pr(\bar{\alpha})_i$. Combining all N sequences gives a refined boundary as:

$$max_{i=1\dots N}\{Pr(\underline{\alpha})_i\} \le Pr(\alpha) \le min_{i=1\dots N}\{Pr(\bar{\alpha})_i\} \quad \dots\dots(14)$$

4.3.5 Memory storage and computational complexity analysis

For memory storage cost, as the message in dPrPm only includes marginal and pairwise node reliabilities, the storage cost is $O(n^2)$.

For computational cost, when adding link $l_{\alpha\to\beta}$ into the intermediate structure, we need only to update the message relating to node β. Thus, to add one link and update the message, at most n nodes are updated. Thus, the computational cost is $O(n)$. In total, there are m links, which

requires updating m times. Thus, the total computational cost is polynomial at $O(mn)$. This is in comparison to typical exponential computational costs for binary networks with n components at $O(2^n)$.

In sum, the workflow of dPrPm is shown in Figure 4.3.5.1.

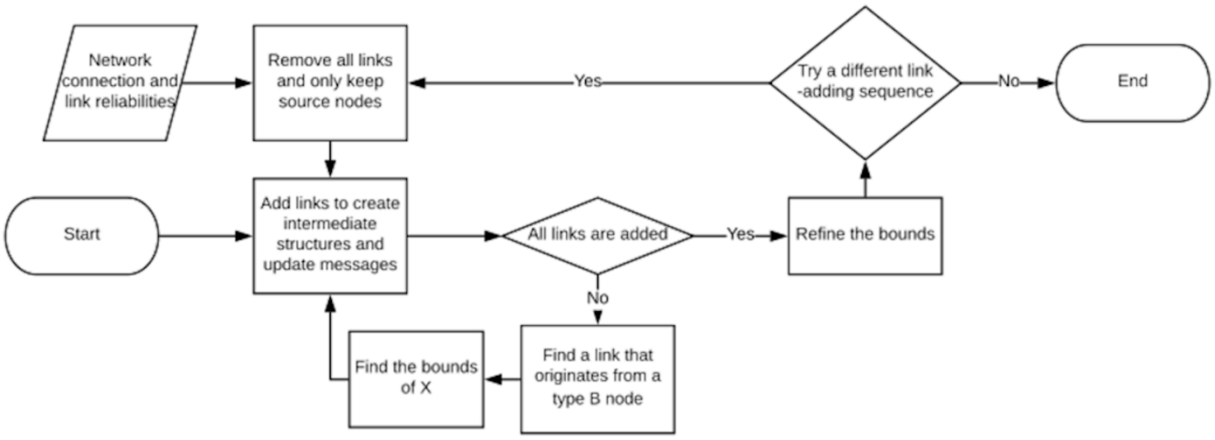

Figure 4.3.5.1 dPrPm workflow

4.4 Test application – directed grid network, power distribution network, gas pipeline network

We now apply the proposed dPrPm to an example network and two real-world applications. We assess the performance of the method in terms of accuracy and computational cost. All results are based on computations run in MATLAB_R2017b on an 8 GB RAM computer. For all three applications, results compared to Monte Carlo simulation are provided. The first two examples also include exact solutions for comparison. A more complex gas pipeline network, where the exact solution is not available, is analyzed in the third example under two seismic scenarios.

Comparisons are made in both accuracy and computational efficiency to assess the performance of dPrPm.

4.4.1 Directed grid network

First, we apply dPrPm to the example network shown in Figure 4.4.1.1, also used for illustration of the method earlier in this chapter. The network is a variation of the undirected grid network that has been studied by Tong and Tien (2018) and Dueñas-Osorio (2017). Instead of looking only at the corner-to-corner reliability of the system, in this example, we change the single-source-single-sink network into a multiple-sources-multiple-sinks directed acyclic network. Nodes 1, 3, and 13 are taken as source nodes and the remaining nodes are sink nodes. The objective is to obtain reliability estimates for the probabilities of being able to receive the resource provided at the source nodes at each of the sink nodes. For simplicity, all links are assumed to be equally reliable with reliability R_l. Varying link reliabilities can be easily incorporated in the probability updating calculations during message passing.

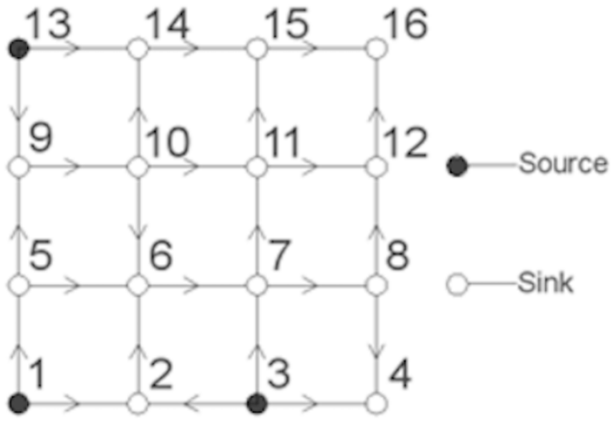

Figure 4.4.1.1 Directed grid network

We test the performance of dPrPm under two cases: a reliable case with $R_l = 0.9$ and an unreliable case where $R_l = 0.1$. Exact solutions are obtained through total enumeration. For dPrPm, we try 100 different link-adding sequences to refine the reliability bounds for each sink node. Percentage error for dPrPm is calculated by taking the median value between the upper and lower bounds. For Monte Carlo simulation, we generate 100000 runs for each node. Results are shown in Table 4.4.1.1 and Table 4.4.1.2. The comparison of computational cost is provided in Table 4.4.1.3.

Table 4.4.1.1 Comparison among exact solutions, Monte Carlo simulation, and dPrPm for directed grid network when $R_l = 0.9$

Node	Exact	Monte Carlo		dPrPm			
		Result	% Error	Lower bound	Upper bound	Gap	% Error
2	0.99000	0.99016	0.01616	0.99000	0.99000	0.00000	0.00000
4	0.98015	0.98022	0.00670	0.98015	0.98015	0.00000	0.00000
5	0.90000	0.89885	-0.12778	0.90000	0.90000	0.00000	0.00000
6	0.99510	0.99475	-0.03545	0.99510	0.99510	0.00000	0.00000
7	0.98956	0.98974	0.01827	0.98956	0.98956	0.00000	0.00000
8	0.89060	0.89086	0.02882	0.89060	0.89060	0.00000	0.00000
9	0.98100	0.98138	0.03874	0.98100	0.98100	0.00000	0.00000
10	0.88290	0.88355	0.07362	0.88290	0.88290	0.00000	0.00000
11	0.97734	0.97758	0.02416	0.97734	0.97734	0.00000	0.00000

Table 4.4.1.1 continued

12	0.97457	0.97456	-0.00054	0.97457	0.97457	0.00000	0.00000
14	0.97946	0.97889	-0.05830	0.97946	0.97946	0.00000	0.00000
15	0.98498	0.98544	0.04671	0.98498	0.98498	0.00000	0.00000
16	0.98539	0.98592	0.05393	0.98506	0.98558	0.00048	-0.00532

Table 4.4.1.2 Comparison among exact solutions, Monte Carlo simulation, and dPrPm for directed grid network when $R_l = 0.1$

Node	Exact	Monte Carlo		dPrPm			
		Result	% Error	Lower bound	Upper bound	Gap	% Error
2	0.19000	0.18899	-0.53158	0.19000	0.19000	0.00000	0.00000
4	0.10092	0.10160	0.66962	0.10092	0.10092	0.00000	0.00000
5	0.10000	0.10191	1.91000	0.10000	0.10000	0.00000	0.00000
6	0.02986	0.03034	1.60529	0.02986	0.02986	0.00000	0.00000
7	0.10269	0.10242	-0.26046	0.10269	0.10269	0.00000	0.00000
8	0.01027	0.01057	2.93370	0.01027	0.01027	0.00000	0.00000
9	0.10900	0.10970	0.64220	0.10900	0.10900	0.00000	0.00000
10	0.01090	0.01169	7.24771	0.01090	0.01090	0.00000	0.00000
11	0.01135	0.01145	0.91177	0.01135	0.01135	0.00000	0.00000
12	0.00215	0.00198	-7.95624	0.00215	0.00215	0.00000	0.00000
14	0.10098	0.09956	-1.40720	0.10098	0.10098	0.00000	0.00000
15	0.01122	0.01177	4.89889	0.01122	0.01122	0.00000	0.00000

Table 4.4.1.2 continued

16	0.00134	0.00114	-14.65189	0.00134	0.00134	0.00000	-0.00041

Table 4.4.1.3 Computational cost comparison among exact solution, Monte Carlo simulation, and dPrPm for directed grid network

	Exact solution	Monte Carlo	dPrPm
Time (sec)	5766.37	29.88	0.32

Compared to the approximated value given by Monte Carlo simulation, dPrPm provides exact upper bounds and lower bounds to the solution. In Table 4.4.1.1 and Table 4.4.1.2, for both the reliable and unreliable cases, dPrPm outperforms Monte Carlo as evidenced by the percentage error values compared to the exact solutions. In addition, in looking at the ability to capture rare events, the reliability at node 16 for $R_l = 0.1$ is the lowest probability event. In Table 4.4.1.2, the percentage error given by Monte Carlo simulation increases to 14.7% for node 16, while the upper and lower bounds given by dPrPm are still narrow. For dPrPm, the percentage error arises from deviations in the calculated reliability beyond the fifth decimal place.

While total enumeration provides the exact solution, it is also computationally intensive. In this example, it takes more than 1.5 hours to generate a solution as shown in Table 4.4.1.3. In comparison, dPrPm takes less than half a second to compute the dPrPm solution 100 times. This is also two orders of magnitude faster than Monte Carlo simulation. Although dPrPm yields an approximated solution with 100% confidence level rather than the exact solution, the gap

between the bounds can be negligible depending on the accuracy requirement and as shown in the results for this example.

4.4.2 Power distribution network

Next, we apply the proposed dPrPm to a four-substation power distribution network from Pacific Gas and Electric (Ostrom 2004) as shown in Figure 4.4.2.1, also investigated in Der Kiureghian and Song (2008) and Tien (2017). The system consists of circuit breakers, switches, and transformers. The solid black circles represent three source nodes and the sink node is node 23. The other 19 components each represent a triplet configuration in the system of switch-breaker-switch.

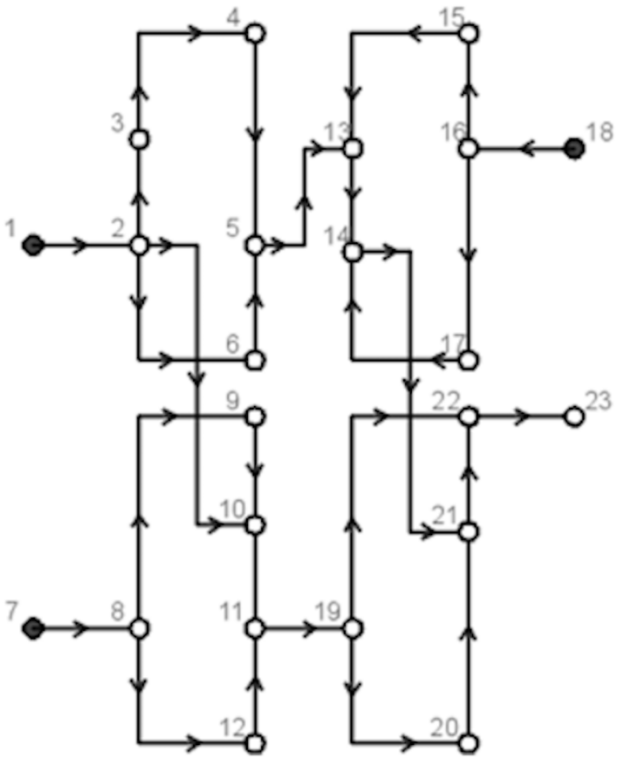

Figure 4.4.2.1 Power distribution network

While dPrPm provides the reliability at all nodes in the network, for simplicity, we only focus on the reliability of node 23 in this example. We assume no nodal failure and each link has the same reliability R_l. Varying link reliabilities can easily be incorporated. For comparison, the method of total enumeration is used to obtain the exact solution. The network is tested under three scenarios for systems of varying reliabilities: $R_l = 0.9,0.99,0.3$. Monte Carlo simulation is also conducted with 100000 realizations for each value of R_l. Results are shown in Table 4.4.2.1. Computational cost comparisons are provided in Table 4.4.2.2.

Table 4.4.2.1 Comparison among exact solutions, Monte Carlo simulation, and dPrPm for power distribution network under different R_l

R_l	Exact	Monte Carlo		dPrPm			
		Result	% Error	Lower bound	Upper bound	Gap	% Error
0.9	0.85741	0.85805	0.07490	0.85591	0.85807	0.00216	-0.04886
0.99	0.98969	0.98939	-0.02981	0.98968	0.98969	0.00001	-0.00018
0.3	0.00221	0.00240	8.63048	0.00221	0.00221	0.00000	0.00295

Table 4.4.2.2 Computational cost comparison among exact solution, Monte Carlo simulation, and dPrPm for power distribution network

	Exact solution	Monte Carlo	dPrPm
Time (sec)	36864.37	35.97	0.36

From Table 4.4.2.1, dPrPm provides a highly accurate solution with maximum percentage error less than 0.05% for all three scenarios. For the lowest probability event, $R_l = 0.3$ in Table 4.4.2.1, Monte Carlo simulation has the highest error at 8.63%, while dPrPm produces a solution with 0.00295% error. In terms of computation time, dPrPm runs almost 100 times faster than Monte Carlo simulation as shown in Table 4.4.2.2.

In this example, we also investigate the effect of using results from multiple link-adding sequences to refine the bounds for dPrPm as given in equation (14). Theoretically, the total number of link-adding sequences satisfying the requirements previously described is finite. As a result, the best solution by dPrPm can be obtained by considering all of them. Figure 4.4.2.2 shows the evolution of the bounds as the number of sequences considered increases. From Figure 4.4.2.2, the gap between the upper bound and lower bound drops quickly in the first 20 runs for all three cases. The size of the gap then levels off. Enumeration of all possible link-adding sequences is computationally intensive. Therefore, rather than finding all possible sequences, we set the number of sequences to 100 to balance accuracy and efficiency.

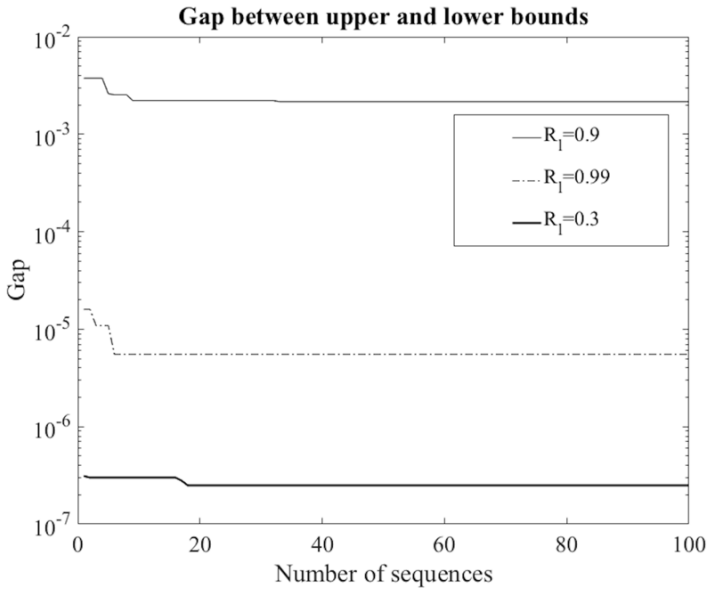

Figure 4.4.2.2 Gap between upper and lower bounds with respect to number of link-adding sequences considered

4.4.3 Gas pipeline network

Finally, dPrPm is tested for a larger and more complex gas pipeline network. The pipeline information for the Los Angeles area is available from the California Energy Commission's GIS open data website. This network has previously been studied to assess reliability of buried pipelines under earthquakes, e.g., Lanzano (2014) and Ambraseys and Menu (1998). The satellite map of the investigated area is shown in Figure 4.4.3.1. The extracted layout of the network is shown in Figure 4.4.3.2, with pipelines between nodes represented by straight lines and arrows indicating the directionality of the links. Here, we analyze the pipeline network reliability under two scenarios: one moderate earthquake with peak ground acceleration (PGA) 0.1g and a more severe earthquake with PGA 0.35g. The network has 96 nodes and 123 directed links. We assume that nodes 1, 4, 20, 92, 36, 46, 47, and 92 are source nodes as marked in Figure

4.4.3.2. The remaining nodes are considered as sink nodes. In total, there are 8 source nodes and 88 sink nodes.

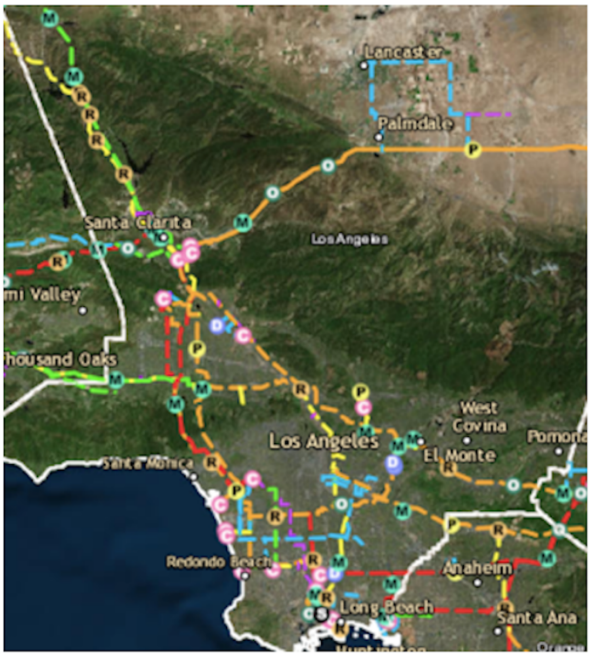

Figure 4.4.3.1 Gas pipeline network (satellite) adapted from California Energy Commission's GIS open data website

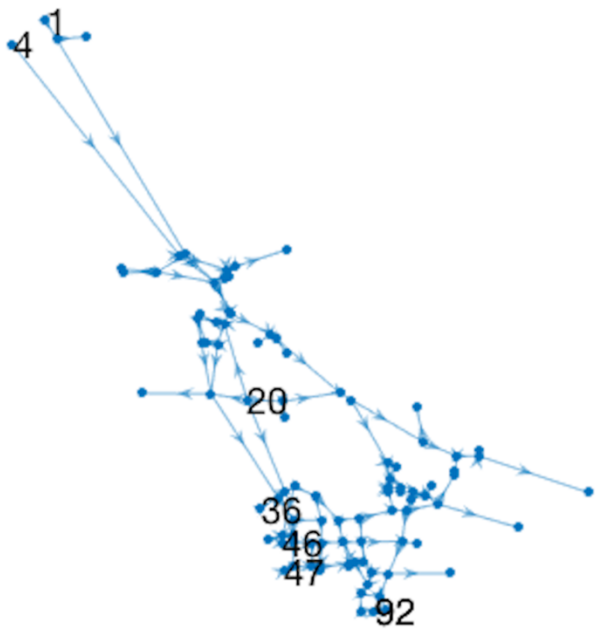

Figure 4.4.3.2 Gas pipeline network (extracted)

In this example, we first find the link reliabilities under the two seismic scenarios as described in Lanzano (2014) and Ambraseys and Menu (1998). Reliabilities at each sink node are then obtained by running dPrPm 100 times and Monte Carlo simulation with 100000 runs under the two earthquakes. The results comparing the two methods are shown in Figure 4.4.3.3 and Figure 4.4.3.4 for PGAs of 0.1g and 0.35g, respectively. For clarity, nodes are ordered by increasing reliability. The exact solution is not available for this network. In all plots, the bold solid line refers to the upper bound given by dPrPm. The thin solid line refers to the lower bound. The dashed line refers to the result given by Monte Carlo simulation. In Figure 4.4.3.3, for a 0.1g earthquake, the upper bound and lower bound overlap, indicating highly confined bounds, and therefore only the upper bound line can be seen. Figure 4.4.3.3(a) and Figure 4.4.3.4(a) show the difference between the dPrPm reliability bounds and results from Monte Carlo. The median value between the upper and lower dPrPm bounds is used as the baseline and the difference from

the median value is shown. Figure 4.4.3.3(b) and Figure 4.4.3.4(b) show the values of the upper and lower bounds by dPrPm and the solution by Monte Carlo simulation.

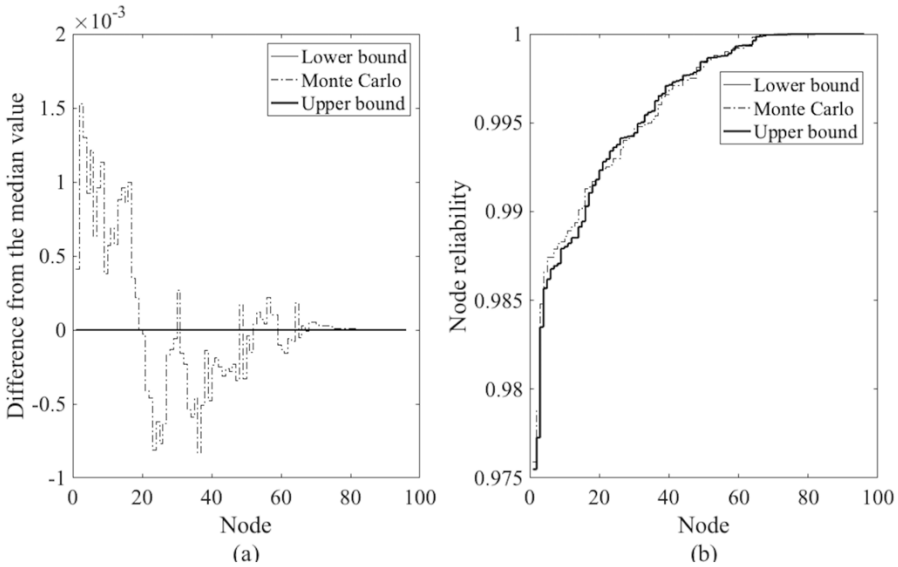

Figure 4.4.3.3 Comparison between dPrPm and Monte Carlo simulation for gas pipeline network under PGA=0.1g (nodes ordered by increasing reliability)

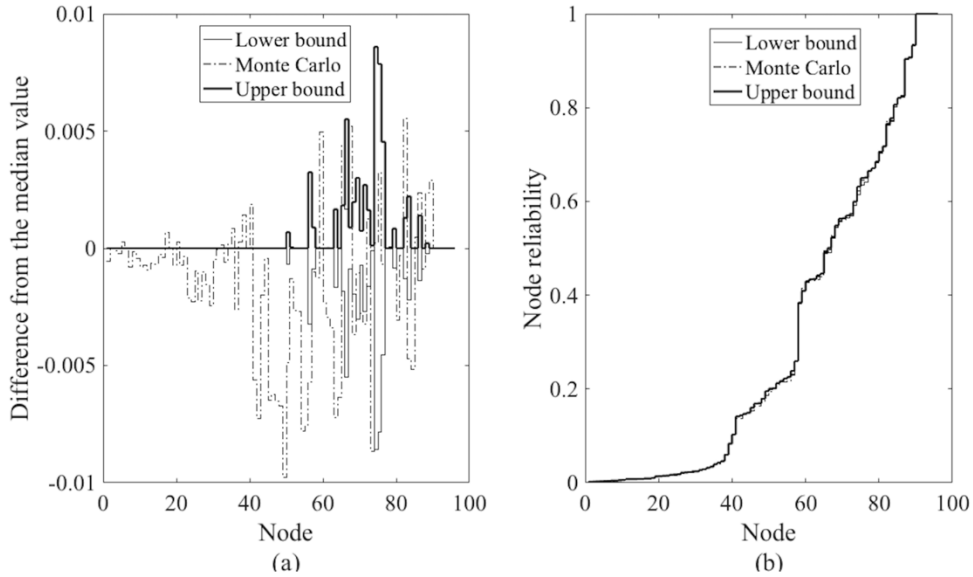

Figure 4.4.3.4 Comparison between dPrPm and Monte Carlo simulation for gas pipeline network under PGA=0.35g (nodes ordered by increasing reliability)

In Figure 4.4.3.3, the narrowness of the bounds obtained by dPrPm is observed. With the guaranteed accuracy of dPrPm, the value of the bounds can be taken as close to the exact solution. In addition, for the approximated results provided by Monte Carlo, it is unknown if the simulations underestimate or overestimate the exact solution, as can be seen by the randomness in the positive or negative differences from the median value in Figure 4.4.3.3(a). In Figure 4.4.3.4, the widest gap between bounds for all nodes is around 1.6%. While dPrPm gives guaranteed upper and lower bounds to the solution, the 100% confidence level is unachievable for Monte Carlo simulation. Table 4.4.3.1 provides the comparison of computation time for

calculation. Consistent with the previous applications, dPrPm takes two orders of magnitude less computation time to obtain a result.

Table 4.4.3.1 Computational cost comparison between Monte Carlo simulation and dPrPm for gas pipeline network

	Monte Carlo	dPrPm
Time (sec)	180.22	1.80

4.4.4 Extension to dependent case and cascading failures

Although dPrPm is designed for networks with conditionally independent links, it can also be extended to dependent cases. One approach to do this is to condition on the parent nodes governing the links. For example, in the gas pipeline network application, links are conditioned on seismic intensity. Given the computational cost is fairly low (in this case, 1.80 sec for a network of 123 links), the prior system reliability can be found by conducting inference over all enumerated parental node combinations if tractable.

The special dependency case where the link reliability $R_{\alpha\to\beta}$ depends on the state of node α is also considered. When link $l_{\alpha\to\beta}$ is added to the intermediate structure, the updating rules shown in Equations (1) to (6) assume the link reliability $R_{\alpha\to\beta}$ is independent of the node state. By replacing $R_{\alpha\to\beta}$ with the link reliability conditioned on the node state ($R_{\alpha\to\beta|\alpha}$), similar updating rules for this special dependent case are built. For example, Equation (3) is modified to

$\overline{Pr(\gamma,\beta)} = Pr(\alpha,\gamma)\, R_{\alpha\to\beta|\alpha=1}$; Equation (6) is modified to $\overline{Pr(\beta,\gamma)} = Pr(\beta,\gamma) + (Pr(\alpha,\gamma) - X)R_{\alpha\to\beta|\alpha=1}$. These modified updating rules enable dPrPm to directly address this case.

In practice, the failures of many networked infrastructure systems are the result of failures propagating or cascading through a network. To capture these effects with dPrPm, we treat a failure as an observation on the system. Based on the observation, we update the link reliabilities accordingly. With the redefined link reliabilities, we then rerun dPrPm to update the reliabilities at all sink nodes. Any new failures that are calculated are the cascaded result from previous node failures.

4.5 Contributions

In this chapter, a new analytical method called the directed probability propagation method (dPrPm) is proposed to evaluate the reliability of acyclic directed networks. Through a defined propagation sequence and accompanying probability updating rules, the method results in guaranteed upper and lower bounds of reliabilities at all sink nodes in a network. The benefits of dPrPm can be summarized in five aspects:

1. No minimum cut sets (MCSs) or minimum link sets (MLSs) are needed to compute network reliabilities. While many other analytical methods rely on MLSs or MCSs, e.g. Bayesian network analysis or recursive decomposition algorithms (RDA), dPrPm does not require the computationally intensive enumeration of component states, MCS, or MLSs to determine the system outcome.

2. The method is applicable to the multiple-sources-multiple-sinks problem. Previous work has investigated node accessibility as a measure of network reliability. However, these methods are often limited to the one-sink problem. To assess the reliability of all sink nodes in the network, researchers have to run the analysis multiple times. In dPrPm, as the message contains the marginal node reliabilities, each run gives the reliabilities of all sink nodes.

3. dPrPm is computationally efficient. Compared to existing analytical algorithms such as RDA and inference in Bayesian networks, computational complexity is reduced from an exponential increase $O(2^n)$ with system size to a polynomial increase $O(mn)$. Time consumption comparisons in the three test applications show the orders of magnitude savings in computation time.

4. Results given by dPrPm are exact bounds. While many other methods, e.g., Monte Carlo simulation, give an approximated answer, 100% confidence level is guaranteed by dPrPm. In many cases, as shown in the test applications, these bounds are narrow, resulting in solutions with very low percentage errors compared to the exact solution.

5. Performance of dPrPm is independent of link reliabilities. Many sampling-based approaches are limited in the ability to analyze rare events or by computational efficiency if the probabilities of rare events are of interest. dPrPm is an analytical method and its efficiency is independent of link or network reliabilities.

Chapter 5 Flow capacity – multistate Bayesian network (BN)

Chapter 5 is organized as follows. We first provide a brief background on the use of BNs for system reliability assessment and describe our BN system formulation. We then present new algorithms for BN modeling of multi-state flow networks and for performing exact inference over these models. These include the proposed approach using compression to reduce the memory storage requirements of the conditional probability tables associated with the BN and two heuristics to increase the computational efficiency of the method. We apply the algorithms to an example infrastructure system to demonstrate their use for reliability assessment. Finally, we assess the performance of the proposed algorithms compared to existing methods in terms of both memory storage and computation time. The corresponding algorithms for compression and inference in multistate Bayesian Network are included in Tong and Tien (2017).

5.1 Introduction – characteristics of Bayesian network (BN)

The Bayesian network (BN) is a useful framework under which to perform infrastructure reliability assessments. For an infrastructure flow network comprised of many interconnected components, the connection between individual components implies the relationship between performance of the overall infrastructure flow network and the state of individual components. Given the uncertainties associated with component performance and the hazards components are subjected to, the BN models component states as random variables and captures the probabilistic dependencies between component and system performance. In addition, in an environment of evolving information, where, e.g., inspections offer new insights into the current states of components, any information entered into the BN propagates through the network to update

assessments of the systems. Finally, the BN as a graphical framework enables transparent modeling of systems to facilitate adoption by end-users.

One of the major challenges in the BN modeling of infrastructure systems is the exponentially increasing computational complexity as the number of components in the system increases.

For a network of N components, the BN model is as shown in Figure 5.1.1. The state of the system, represented as a system node Sys, is dependent on the states of each of its constituent components, represented as component nodes $C_1, C_2, \dots, C_n$. In BN terminology, $C_1, C_2, \dots, C_n$ are called parents of Sys.

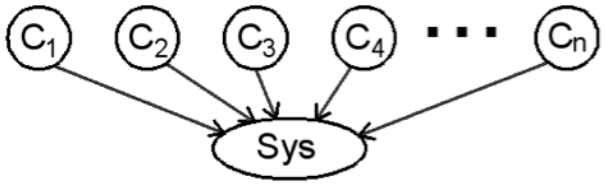

Figure 5.1.1 BN model of a system comprising n components

The BN is a probabilistic graphical model. For the BN, each node must be associated with a conditional probability table (CPT), which gives the probability distribution of that node given each of the mutually exclusive combinations of states of its parents. Nodes that do not depend on other nodes are defined by marginal probability distributions. The reader is referred to texts such as Jensen and Nielsen (2007) for further details on BNs.

Now suppose that the components and system can be in one of multiple possible states, e.g., states 0, 1, 2, 3, or 4 denoting discretized values of flow capacity 0%, 25%, 50%, 75%, and 100% of maximum capacity, respectively. An example of the CPT associated with the system node for this multi-state flow network is shown in Table 5.1.1. Let n denote the number of components and m the number of states of each component. For the columns indicating system states, we use m columns to represent whether the system is in that specific state. If so, the value in the column is 1; otherwise, it is 0.

Table 5.1.1 Example conditional probability table for a multi-state system

C_1	⋯	C_{n-1}	C_n	$sys = 0$	$sys = 1$	$sys = 2$	…	$sys = m - 1$
0	⋯	0	0	1	0	0	…	0
0	⋯	0	1	1	0	0	…	0
⋮	⋮	⋮	⋮				⋮	
4	⋯	4	0	0	1	0	…	0
4	⋯	4	1	0	0	1	…	0
4	⋯	4	2	0	0	0	…	1
4	⋯	4	3	0	0	0	…	0
4	⋯	4	4	0	0	0	…	0

As the number of components in the system increases, the size of the CPT as shown in Table 5.1.1 increases exponentially. In general, the system states in the CPT are represented by

$m \times m^n$ elements, as shown in the m rightmost columns in Table 5.1.1. Due to mutually exclusive system states, knowing the column in which the 1 value appears, one can infer the values in the other columns for that row. This reduces the number of elements to m^n. However, this value is still exponentially increasing with the number of components in the system being modeled. For a 5-state system of $n = 100$ components, for example, the full representation of the CPT includes a minimum of $5^{100} = 7.9 \times 10^{69}$ elements. The exponential increase in the size of the CPT poses a significant memory storage challenge in constructing and analyzing the BN. It quickly renders the model intractable, necessitating the development of new methods to enable the BN modeling of larger multi-state flow systems.

5.2 Literature review – Applications of reliability analysis based on BN

Assessing the impact of component performance on system performance enables a stakeholder to identify the most critical components, and prioritize decisions for inspection, repair, or replacement of these system elements. The objective is to create more reliable systems under both normal operating and hazard conditions, leading to improved community outcomes (Johansen et al. 2016).

To tackle the challenge in dimensionality, storage and computationally efficiency in applying BN. Previously, algorithms were proposed for BN modeling of binary systems (Tien and Der Kiureghian 2013) to enable the study of larger infrastructure systems within the BN framework (Tien and Der Kiureghian 2016). However, this binary system formulation, where components and the system are in one of two states, e.g., failure of survival, is not sufficient to describe the status of many infrastructure components and systems (Tong and Tien 2016). This is true for

infrastructures including water, gas line, and transportation networks, which are characterized by flow, e.g., of water supply, natural gas, and vehicles, across the network. In these cases, if a component is functioning at, e.g., 50% of its maximum capacity, this cannot be sufficiently defined as failure or survival. Compared with binary systems, multi-state system modeling provides a more detailed description of system reliability and enables the analysis of flow instead of connectivity networks. The dimensionality of the problem, however, also increases.

Previous studies on the use of BNs for modeling system performance have focused on generating BNs from conventional system modeling methods, such as reliability block diagrams (Torres-Toledano and Succar 1998; Kim 2011) and fault trees (Bobbio et al. 2001). These and other studies using BNs for system reliability assessment (Mahadevan et al. 2001; Boudali and Dugan 2005; Khakzad et al. 2011; Tien and Der Kiureghian 2015) have all assumed binary component and system states. This allows the modeling of systems characterized by connectivity, such as the power distribution network in Tien and Der Kiureghian (2017), but not flows.

In the study of multi-state systems, Bouissou and Pourret (2003) propose a BN-based method for performance evaluation of systems with multiple states. The focus, however, is on troubleshooting, or identification of single causes of system failure. The assumption of single-fault failures does not hold in the assessment of civil infrastructure systems. The reduced capacity of multiple components may lead to reduction in system performance. Gu and Yang (2013) use BNs to assess the reliability of multi-state systems. The system studied, however, consists of only six components, and even with this small number of components, intermediate

nodes are introduced in the BN to enable computationally tractable calculations using the traditional BN modeling method.

Bensi et al. (2013) propose a method for more efficient modeling of BNs, including for multi-state systems. The study takes a topology optimization approach to address the system size limitation of BN models. The optimization algorithm, however, must consider all permutations of the indices of system components, and therefore may itself become intractably large for large infrastructure systems. Finally, a method based on the flow conservation law is proposed in Yeh (2013) to assess the reliability of a multi-state flow network. The focus, however, is on modeling deterioration effects, rather than on quantifying the importance of individual component performance on overall system reliability. In addition, the systems considered in this paper do not necessarily obey the flow conservation law, i.e., that the flow into any node is equal to the flow out of that node. For example, in our systems, a given level of flow may enter a system component, but given the state of that component, the flow out may be different. Thus, new methods are required for the BN modeling of multi-state networks.

5.3 Theoretical deduction

5.3.1 Workflow

The flowchart of the proposed algorithms is shown in Figure 5.3.1.1. Each of the modeling and analysis steps is described in detail in the following sections. As an overview, first, to reduce the size of the CPT, we substitute subsystems with components that are either in series or parallel as super-components. Next, we generate the minimum cut sets (MCSs) of the network to determine

the system state for each combination of component states. To facilitate efficiency of the compression algorithm that follows, we renumber the super-components that now comprise the full system based on two heuristics: whether they may be observed and their appearances in the MCSs. This completes the formulation of the system for the BN model.

Next, as shown in Figure 5.3.1.1, to construct and perform inference over the model, compression and preprocessing algorithms are proposed. These algorithms are run simultaneously to reduce computational time. To compress the system node CPT, we introduce the idea of bundles representing repeated patterns of fixed length in the system state in the CPT. The simultaneous preprocessing of the information in the BN removes the need for storage of intermediate factors of unobserved components during inference, reducing both memory storage requirements and computational time. At the end of the algorithm, we obtain a compressed system CPT, $cCPT$, and dictionary of bundles, d, used in the compression. These contain all the information for the BN model of the system compressed in a lossless manner and without making any approximations. The prior probability distribution for the system before any observation is made, λ_1, and subsequent distribution and dictionary λ_{i+1} and d_{i+1}, respectively, are also obtained for inference on posterior probability distributions given observations on the system. Each of the elements of the proposed method is now described in detail.

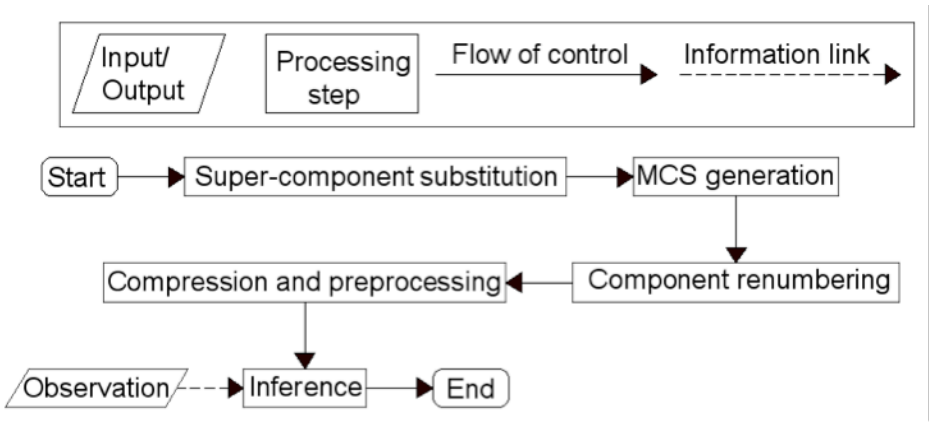

Figure 5.2 Flowchart of proposed algorithms for BN modeling of multi-state systems

5.3.2 Super-component

A common practice in the field of reliability analysis is to combine components in similar configurations into a single component, which we call a super-component and denote as C_c. The probability distribution of the ith super-component C_{ci} is denoted as $p(C_{ci})$. Two most recognized configurations are shown in Figure 5.3.2.1, with components in series (left) or parallel (right) subsystems. These simple configurations are widely used in civil infrastructure systems, including redundant components arranged in parallel in mesh-type networks and components in series in linear pipeline systems. Instead of including all individual components in a series or parallel subsystem in the CPT, we use one super-component with updated probability distributions to represent the subsystem. For consistency in the formulation, a component that does not belong to a series or parallel subsystem is treated as a super-component on its own. The probability distribution of a super-component can be easily determined by the characteristic of a series or parallel system. In chapter 5, we continually reduce the complexity of the network using super-components until no two components in the system remain in series or parallel.

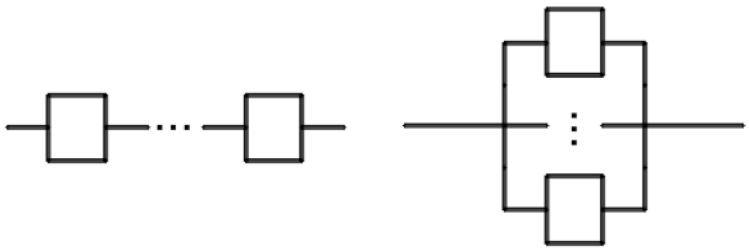

Figure 5.3.2.1 Super-component configurations

One advantage of using super-components is that if additional components are added to the structure of previously defined super-components, this will not increase the size of the overall problem because the number of super-components is not changed. Only the probability distribution of that super-component need be modified. Of course, more complicatedly connected subsystems other than series and parallel configurations can be defined as super-components. This may further simplify the structure of the system for specific cases, particularly if there are several repetitions of a similar structure in the network. Here, without loss of generality, we limit the discussion of super-components to series and parallel subsystems.

5.3.3 Generate MCS

In our BN formulation, determination of the system state in any row of the CPT is based on the component states and the set of minimum cut sets (MCSs) or minimum link sets (MLSs) of the system. The component states are determined by their row number in the CPT as in Tien and Der Kiureghian (2016). Enumerating all MCSs or MLSs for a system is an NP-hard problem, though several efficient methods have been developed to do so (Suh and Chang 2000; Li et al. 2007; Benaddy and Wakrim 2012). The EG-CUT algorithm proposed by Shin and Koh (1998) generates MCSs for undirected graphs by using a blocking mechanism. MCSs can be generated

at $O(en)$, where e is the number of edges and n the number of nodes in the graph. Depth-first search-based methods can also be used for MLS generation (Jiang et al., 2016). These have been used to find the MLSs for an infrastructure system of 127 nodes using a personal computer on the order of seconds (Johansen and Tien 2017).

5.3.4 Renumber

Once the MCSs have been generated, we propose two heuristics for numbering the components to further reduce the computational complexity of the problem. The number of a component affects where it appears in the system CPT, i.e., its column in the lefthand side of the CPT as shown in Table 5.1.1. The heuristics proposed to renumber the components result in reduced memory storage required for inference and more efficient compression of the initial system CPT. In general, selection of an optimal component numbering in the network is an NP-hard problem (Dechter 1999). In our method, our first priority is to number the components that may be observed to appear as far left in the CPT as possible. When the state of a component is observed, the probability distribution of the system node is updated given that information using Bayes' rule. In most current infrastructure systems, particularly water and gas line infrastructure, not all components have monitoring capabilities. This heuristic takes advantage of this system characteristic to reduce the size of the intermediate probability distributions, denoted λ and called intermediate factors, calculated during the inference process. For the variable elimination inference method used in this paper, when a component is observed, only the part of λ that corresponds with the component being in the observed state need be considered. Numbering the monitored components first to appear on the left side of the CPT reduces the storage required for λ.

To facilitate more efficient compression of the system CPT using the algorithm described in the following section, the second part of the renumbering heuristic is to number components with greater influence in affecting the system state to appear to the left in the CPT (after the monitored components previously described). Appearance in MCSs is used as a proxy for component influence and criticality (Meng 1994). Specifically, after obtaining the MCSs, we rank the influence of each component by its number of appearances in a MCS. The most influential component is the one that appears in the most MCSs. By renumbering components based on influence so that more influential components have smaller component numbers results in a CPT where the system state does not change rapidly from row to row in the CPT. This results in a more regular pattern that improves the computational efficiency of compression.

5.3.5 Compress

With the BN model created after super-component substitution and renumbering, the number of parents of the system node in Figure 5.1.1 is reduced to n_c, where n_c is the number of super-components. In constructing the CPT, we use the super-components instead of original components. Now, rather than storing all elements in the full CPT, we propose an algorithm to compress the information in the CPT to reduce the memory storage requirements and make the BN modeling of larger civil infrastructure systems possible. First, the component states in the CPT (lefthand columns in Table 5.1.1) need not be stored, as long as they follow a predetermined pattern. Next, compression is accomplished by processing through each row of the full system CPT and storing the values that appear in the system state columns (righthand columns in Table 5.1.1) in a lossless compressed form. To do this, we introduce the idea of

bundles. A bundle is a pattern in the values of the system state of fixed length that is proportional to the number of states. These are more specific than the general *phrases* terminology originally proposed by Ziv and Lempel (1977). However, consistent with previous work, we store identified bundles in a *dictionary*. The compressed CPT therefore comprises detected bundles – patterns of sequences in the system columns of the CPT – which are referenced from the dictionary.

The idea of bundles is important to remove the need to calculate certain *remainder* values when processing through the CPT as in the previously developed algorithms for BN modeling of binary systems. The reader is referred to Tien (2014) for details in the calculation of this remainder. As the number of possible states of the system increases, so does the likelihood of having remainders. Employing bundles removes the possibility of remainders. Let m denote the number of states of the components and system. With fixed-length bundles of lengths that are multiples of m stored in the dictionary rather than allowing phrases of general length, the CPT is thus compressed in groups proportional to m, removing any remainders from the compression process.

The proposed compression algorithm operates as follows. The outputs of the algorithm are the compressed system CPT, $cCPT$, and the accompanying dictionary of bundles, d. Examples of the algorithm are shown in Figures 5.3.5.1 and 5.3.5.2.

Dictionary=

Bundle number	1	2
Bundle	{0,0,0}	{0,1,1}

cCPT=

Bundle number	1	2	1	2	1	2	1	...
# repetitions of bundle	312	2	1	2	1	2	1	...

Figure 5.3.5.1 Illustration of example 3-state system CPT compression

Dictionary=

Bundle number	1	2	3
Bundle	{0,0,0}	{0,1,1}	{0,1,1,0,1,1,0,0,0}

cCPT=

Bundle number	1	3	...
# repetitions of bundle	312	100	...

Figure 5.3.5.2 Illustration of 3-state system CPT compression after combination of phrases

The full flowchart of the algorithm is shown in Figure 5.3.5.3.

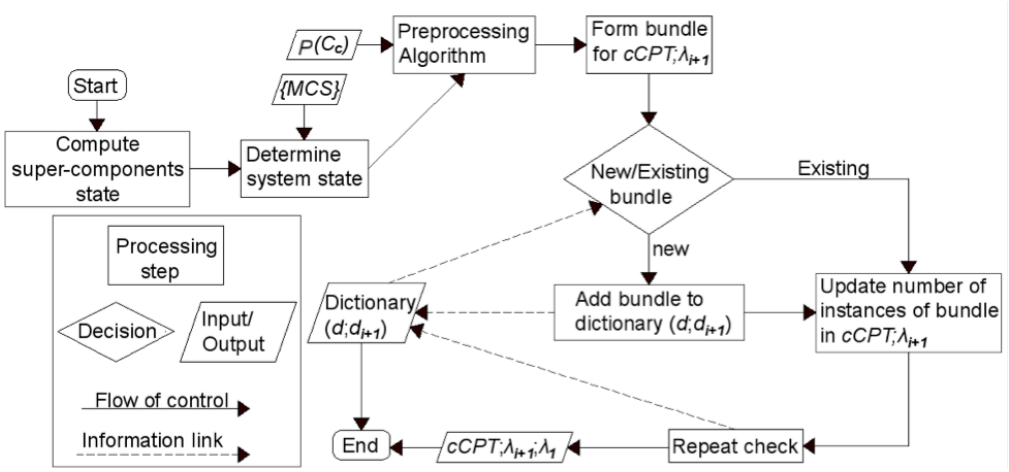

Figure 5.3.5.3 Flowchart of compression algorithm

5.3.6 Construct BN

For each row $n_r = 1,2,\dots,m^{n_c}$ of the system CPT, the states of the super-components $S_1,\dots,S_{n_c}$ represented in that row are computed based on the specific pattern used in defining the CPT. The CPT, as shown in Table 5.1.1, is constructed with super-components $C_{c1},\dots,C_{cn_c}$ organized from left to right. Each row of the CPT is one of the mutually exclusive combinations of component states. We determine the state of component $C_{ci}, i = 1,\dots,n_c$ in row n_r of the CPT according to equation (1)

$$S_i = \mathrm{mod}\left(\mathrm{ceil}\left(\frac{n_r}{m^{n_c-i}}\right) - 1, m\right) \qquad (1)$$

where $\mathrm{ceil}(x)$ is the value of x rounded up to the nearest integer; $\mathrm{mod}(a, m)$ returns the remainder after division of a by m. The possible states of the component $C_{ci} \in \{0,1, \dots, m-1\}$.

For each row, the component states are then checked against $\{MCS\}$ (indicating the set of minimum cut sets of the system) by equation (2) to determine the state of the system in row n_r, denoted sys_{n_r}.

$$sys_{n_r} = \min_{j=1,\dots,n_{MCS}} \left\{\max\left\{MCS_{j,n_r}\right\}\right\} \qquad (2)$$

n_{MCS} denotes the number of MCSs of the system, and any one $MCS_{j,n_r}, j = 1, \dots, n_{MCS}$ contains the states of the components comprising that MCS in row n_r of the CPT. From the value of sys_{n_r}, we obtain the corresponding binary values in the m entries for the system state in row n_r.

In addition to storing phrases as bundles of fixed length proportional to the number of states m, we have found that additional computational efficiencies can be achieved by combining multiple bundles. This is particularly effective if there are repeated patterns in the occurrence of bundles. For example, consider a 3-state system with dictionary and compressed system node CPT represented by $cCPT$ as shown in Figure 5.3.5.1. In this case, the pattern in bundles in $cCPT$ is represented by 2 repetitions of bundle 2, then 1 repetition of bundle 1. The memory storage requirements for the compressed CPT can be reduced significantly by defining a new bundle 3 comprising the bundle $\{0,1,1,0,1,1,0,0,0\}$ as shown in Figure 5.3.5.2. We call this new, longer bundle a *clip*. Specifically, the original memory storage requirement is

$(\text{memory for dictionary}) + (\text{memory for cCPT}) = (3 + 3) + (201 \times 2) = 408$ elements, while combining the bundles into a clip results in a memory storage requirement of $(3 + 3 + 9) + (2 \times 2) = 19$ elements. Though the size of the dictionary slightly increases with the longer-length clip, the size of the compressed CPT is significantly reduced. Note that combining bundles is not beneficial in all cases. For example, if there are 50 repetitions of bundle 2 followed by 50 repetitions of bundle 1, the length of the clip itself would be $(50 + 50) \times 3 = 300$ elements.

To identify repeated patterns among bundles to combine them into clips, we process through $cCPT$ from left to right through the columns of $cCPT$. For improved efficiency of repeated pattern finding, we limit the length of the clip to the number of states, i.e., for a 3-state system, we do not look for a repeated pattern over the length of 3. Performing this repeat check multiple times, i.e., combining repeated clips into new, longer clips, was also considered. However, this required additional computational time and did not result in savings in memory storage. Therefore, the repeat check algorithm is run through the data once for full compression.

5.3.7 Preprocess

Once the BN has been constructed, inference is required to draw conclusions about the system. In the proposed method, a preprocessing algorithm is used as a prerequisite for inference. This is done simultaneously with the compression algorithm to eliminate the need to run through the data twice and to further reduce memory storage requirements. The preprocessing algorithm is based on the classical variable elimination algorithm (Dechter 1999). In this algorithm, inference is achieved by eliminating all other nodes in the network until we arrive at the node of interest.

Elimination of each node corresponds to summing of the joint distribution over all states of the node, resulting in an intermediate factor λ that is used during the next step of elimination.

Consider n_c super-components of a system $C_{c1}, \dots, C_{cn_c}$ with probability distributions $p(C_{c1}), \dots, p(C_{cn_c})$. If the maximum super-component number of the monitored components is i, then the intermediate factor of interest is λ_{i+1}. In a backwards elimination order of $k = \{n_c, n_{c-1}, \dots, i+1\}$, the elimination of a component k results in an intermediate factor λ_k. At each elimination step, we multiply the values of the elements in λ_k with the probability distribution of the component we are eliminating to obtain λ_{k-1} in the next step. This algorithm results in exact inference over the network.

The key to computationally tractable exact inference in BNs with many parent nodes using the proposed compression approach is that the intermediate factor λ_{i+1} is also stored in compressed form following the same methodology used to create $cCPT$. This is done using the proposed preprocessing algorithm that is run simultaneously with the compression algorithm. Preprocessing is possible because the full intermediate factor does not have to be constructed for the variable elimination method to proceed. For example, as we process through the CPT from the first row to the mth row, the first entry of λ_{n_c} is already determined. The following rows will not affect the value of that first entry and after calculating the value, the first mth entries can be cleared from storage. We need not present at any time the full λ_{n_c}. Similarly, once the mth entry is stored in λ_k, the next entry for λ_{k-1} can be determined. Since the intermediate factor of interest is λ_{i+1}, for all other intermediate factors, storage of only m entries for each λ is needed at a time.

Let e_k denote the values currently stored in λ_k. When the n_rth row is processed, the updating rule for intermediate factors $\lambda_k, k > i+1$ is to use the m entries in e_k and $p(C_{ck-1})$ to create the next entry for e_{k-1}. This is done with the calculation shown in in equation (3)

$$e_k' = e_k[P(C_{ck} = 0) \quad \cdots \quad P(C_{ck} = m-1)]^T \qquad (3)$$

where $e_k{}'$ indicates the next entry for λ_{k-1}. The contents in e_k can now be cleared. Once we arrive at λ_{i+1}, all entries are stored for future use in inference using the compression methodology. Thus, CPT and λ_{i+1} are compressed simultaneously. The algorithms operate as follows.

Preprocessing Algorithm

Input: n_c, m, $p(C_c)$, i

Output: λ_{i+1},d_{i+1}, λ_1

For n_r from 1 to m^{n_c}

Determine system state for row number n_r by equations (1) and (2).

Set $k = n_c$, $N = floor\left(\frac{n_r}{m}\right)$, $R = mod(n_r, m)$

While $R = 0$

Update e_k to $e_k{}'$ based on $p(C_{ck})$ from $p(C_c)$ by equation (3).

Update contents in e_k and e_{k-1}.

Update $k = k-1$, $N = floor\left(\frac{N}{m}\right)$, $R = mod(N, m)$

End while

Compress entries in λ_{i+1} and companion dictionary d_{i+1} using the compression algorithm.

End for

$p(sys) = \lambda_1$

Compression Algorithm

Input: n_c, m, $\{MCS\}$, $p(C_c)$, i

Output: $cCPT$, d, λ_{i+1}, d_{i+1}, λ_1

For $n_r \leftarrow 1$ to m^{n_c}, do as shown in Figure 5.3.5.3.

5.4 Test application

5.4.1 Seven-component network and its variation form

We now apply the proposed algorithms to a test application to illustrate their use. The example system is adopted from Bensi et al. (2013) with added complexity to demonstrate the methodology. The system has previously been used as an example of an infrastructure distribution network, providing a resource, e.g., water or gas, from a source to a sink as in Tien and Der Kiureghian (2015). In this section, we begin with this network as shown in Figure 5.4.1.1.

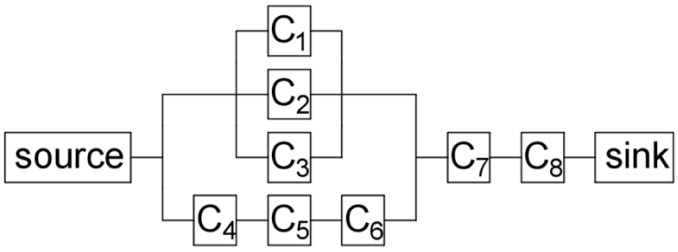

Figure 5.4.1.1 Super-component C_{ci} configuration

However, because of the combination of series and parallel configurations of components C_1 to C_8, it can be reduced into a single super-component. Therefore, we call this system super-component C_{ci} and increase the complexity of the system to form the network shown in Figure 5.4.1.2. This network is formed as a combination of subsystems. Representing the network in Figure 5.4.1.2 using super-components results in the system shown in Figure 5.4.1.3, with each super-component $C_{ci}, i = 1, \dots, 7$ as shown in Figure 5.4.1.1. Thus, the full network of 56 components is represented with 7 super-components. Compared to the system in Figure 5.4.1.1 that can be reduced to one super-component, the network shown in Figure 5.4.1.3 is irreducible using the described super-component methodology. Therefore, this network is chosen to test performance of the algorithms.

Suppose all components are independent and can be in one of 5 possible states modeling the level of flow compared to maximum flow capacity. Let state $S_i = \{0,1,2,3,4\}$ denote flow=0%, 25%, 50%, 75%, and 100% of maximum capacity. Prior probability distributions for each component are listed in Table 5.4.1.1, where S_i denotes the state of component C_i.

Table 5.4.1.1 Prior probability distributions for components constituting super-component C_{ci}

	$p(C_1)$	$p(C_2)$	$p(C_3)$	$p(C_4)$	$p(C_5)$	$p(C_6)$	$p(C_7)$	$p(C_8)$
$S_i = 0$	0.18	0.16	0.14	0.12	0.10	0.08	0.06	0.04
$S_i = 1$	0.19	0.18	0.17	0.16	0.15	0.14	0.13	0.12
$S_i = 2$	0.20	0.20	0.20	0.20	0.20	0.20	0.20	0.20
$S_i = 3$	0.21	0.22	0.23	0.24	0.25	0.26	0.27	0.28

Table 5.4.1.1 continued

$S_i = 4$	0.22	0.24	0.26	0.28	0.30	0.32	0.34	0.36

The resulting prior distribution for each super-component C_{ci} is listed in Table 5.4.1.2.

Table 5.4.1.2. Prior probability distribution for super-component C_{ci}

	$S_i = 0$	$S_i = 1$	$S_i = 2$	$S_i = 3$	$S_i = 4$
$p(C_{ci})$	0.0986	0.2364	0.3257	0.2692	0.0701

Suppose components C_1 of C_{c1} and C_{c2} are instrumented and can be monitored for updating of the system state. For the reduced system shown in Figure 5.4.1.3, the set of MCSs is $\{MCS\} = \{(C_{c1}), (C_{c7}), (C_{c2}, C_{c3}), (C_{c5}, C_{c6}), (C_{c2}, C_{c4}, C_{c6}), (C_{c3}, C_{c4}, C_{c5})\}$.

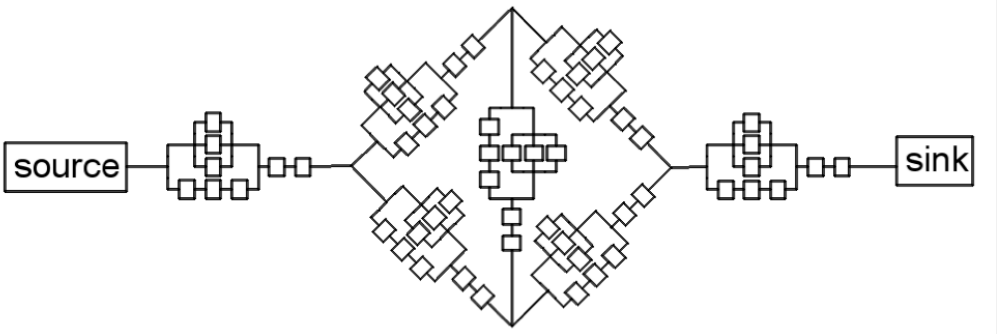

Figure 5.4.1.2 Example system

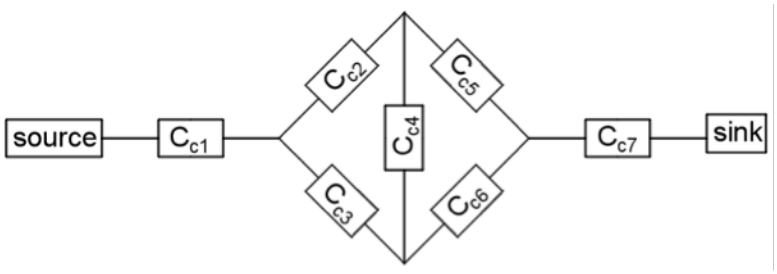

Figure 5.4.1.3 Example system with super-component representation

Compression results

To implement the compression and preprocessing algorithms, for each row in the system CPT, the states of the components are first determined by equation (1). The state of the system is then found using MCSs as shown in equation (2). The resulting dictionary, d, and compressed system CPT, $cCPT$, obtained after implementing the compression algorithm are shown in Tables 5.4.1.3 and 5.4.1.4, respectively.

Table 5.4.1.3 Dictionary for $cCPT$: d

Bundle Number	1	2
Bundle	(0,0,0,0,0)	(1,1,1,1,1)
Bundle Number	3	4
Bundle	(1,0,0,0,0)	(1,0,0,0,0,1,0,0 …,1,1,1,1,1)
Bundle Number	5	
Bundle	(1,0,0,0,0,1,0,0 …,1,1,1,1,1)	

Table 5.4.1.4 $cCPT$

Bundle Number	2	5	4	5	...	2	5	3
# repetitions of bundle	3251	4	5	4	...	1	23	24

After compression, there are a total of 5 bundles identified and $cCPT$ contains 120 columns. It is noted that bundle #4 is a clip combining 4 repetitions of bundle #3 and 1 repetition of bundle #2; bundle #5 combines 24 repetitions of bundle #3 and 1 repetition of bundle #2. Compared to 78125 rows of the full CPT, we see that the compressed form comprising 2025 total elements achieves orders of magnitude savings in memory storage for the CPT. This compression is done in a lossless manner and without making any approximations.

While we implement the compression algorithm for the system CPT, $P(sys = 0) = 0.2045$ is calculated simultaneously using the preprocessing algorithm. The probability distribution over the system state will be updated if the monitored components are identified to be in a specific state. In this case, if components C_1 of C_{c1} and C_{c2} are monitored, then the intermediate factor of interest is λ_3. This is created simultaneously using the preprocessing algorithm as we compress the original CPT. d_3 and the compressed λ_3 denoted $c\lambda_3$ are as listed in Tables 5.4.1.5 and 5.4.1.6.

Table 5.4.1.5 Dictionary for $c\lambda_3$: d_3

Bundle Number	1	2
Bundle	(1,1,1,1,1)	(0.2025,0.1081,0.1081,0.1081,0.1081)

Table 5.4.1.6 $c\lambda_3$

Bundle Number	1	2
# repetitions of bundle	1	4

Inference

Once we obtain d_3 and $c\lambda_3$, we are able to conduct exact inference over the network. Suppose we observe that C_1 in both C_{c1} and C_{c2} are in state 0. We then update the prior distribution of C_1 as $P(C_1 = 0) = 1$. For all other states $m = 1, \dots, 4$, $P(C_1 = m) = 0$. As a result, the probability distribution for super-components C_{c1} and C_{c2} are updated as: $P(C_c = 0) = 0.1031$, $P(C_c = 1) = 0.2580$, $P(C_c = 2) = 0.3382$, $P(C_c = 3) = 0.2453$, $P(C_c = 4) = 0.0554$. We then use the new C_c distributions with λ_3 to obtain the updated system state distribution $P(sys = 0 | C_1$ in C_{C1} and $C_{C2} = 0) = 0.2088$. This represents a slight increase in the probability of failure from 0.2045. This is due to C_1 being one component of a parallel subsystem within only two super-components. In general, once the compressed $c\lambda_{i+1}$ is obtained from the preprocessing algorithm, the inference is computed using simple calculations based on $c\lambda_{i+1}$.

In many system reliability problems, an observation at the system level is made, and the objective is to identify the components most likely to have led to that system behavior. This is called backward inference. Part of the power of BNs is in its ability to perform backward inference by Bayes' rule: $p(C|C_{ci}) = \frac{p(C_{ci}|C)p(C)}{p(C_{ci})}$ and $p(C_{ci}|sys) = \frac{p(sys|C_{ci})p(C_{ci})}{p(sys)}$. An example of the results of this inference is given in Figure 5.4.1.4, which shows the updated probability distributions of each component being in different states given a super-component C_{ci} being in

state 2. Figure 5.4.1.5 shows the updated probability distributions of each super-component being in different states given the system being in state 2.

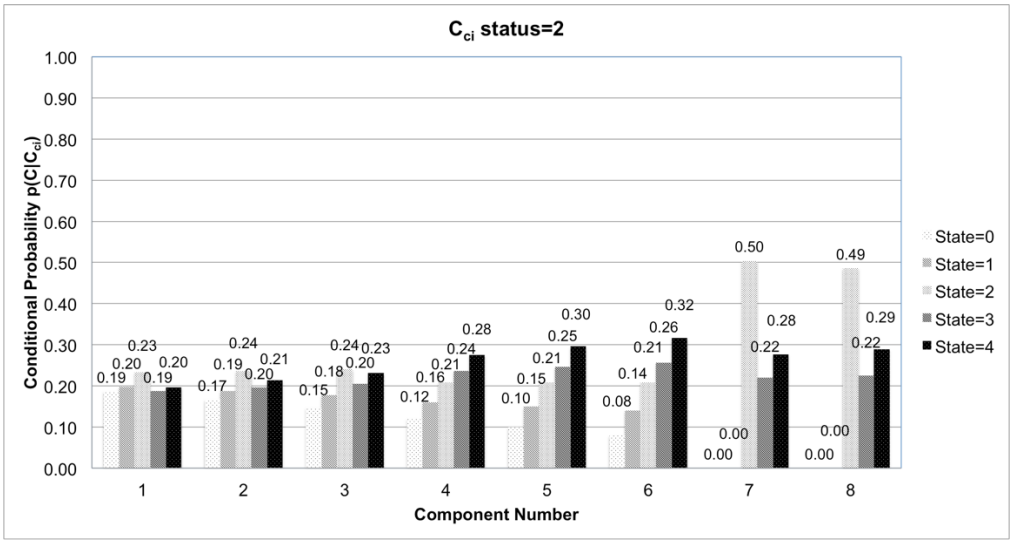

Figure 5.4.1.4 Updated component probability distributions given a super-component C_c in state 2

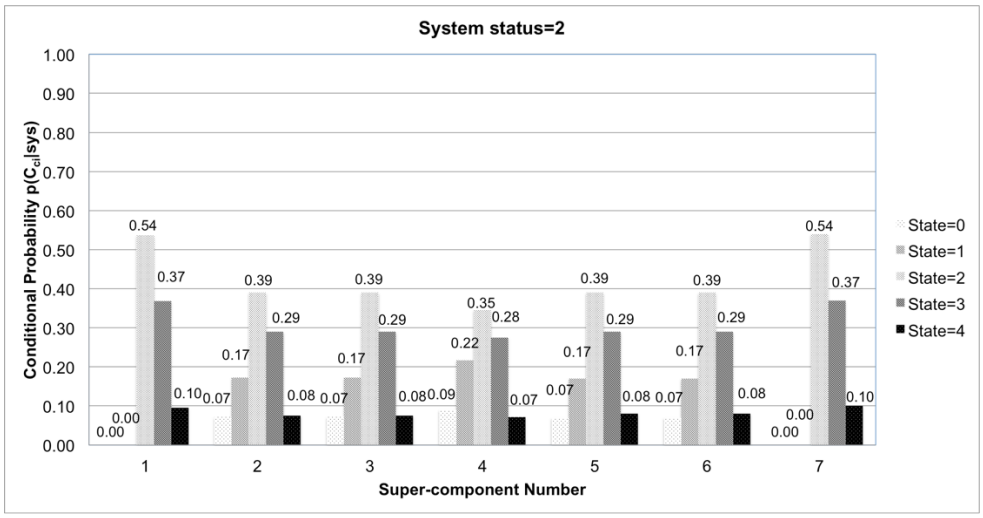

Figure 5.4.1.5 Updated super-component probability distributions given system in state 2

When we have evidence of underperformance at the system level, the results in Figure 5.4.1.4 provide information on the importance of individual super-components comprising the system. Using the chain rule and total probability, the influence of specific components constituting the super-components is determined by $p(C|sys) = \sum_{C_{ci}} p(C|C_{ci}, sys)p(C_{ci}|sys) = \sum_{C_{ci}} p(C|C_{ci})p(C_{ci}|sys)$. These inference results enable identification of critical components in the system to inform decision-making in the maintenance and reinforcement of the critical components to minimize risk of system underperformance.

Storage comparison

In the previous section, we applied the proposed algorithms to an example system to demonstrate their use. As the objective of the algorithms is to reduce the memory storage requirements and increase the computational efficiency of BN modeling of multi-state flow networks as the number of components in the system increases, we now compare the performance of the proposed algorithms with existing methods for modeling systems of increasing size.

First, we assess performance in terms of memory storage. The existing method for comparison is the junction tree (JT) algorithm as implemented in the Bayes Net Toolbox in Matlab (Murphy 2001). The JT algorithm is generally known for its efficiency in performing exact inference for graphical networks (Spiegelhalter et al 1993). We increase the size of the system by adding to the last super-component components in parallel up to a total number of super-components n as shown in Figure 5.4.1.6. As the purpose is to analyze the effect on compression of increasing the number of components, super-components C_{c7} to C_{cn} are not combined into a single super-

component. Similar analyses can be conducted for increasing the number of components elsewhere in the system.

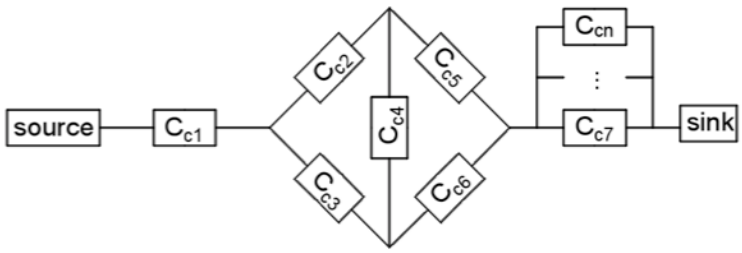

Figure 5.4.1.6 Expanded example system

The maximum number of elements required to be stored to create the BN model and for inference is used as a proxy for memory storage demand. Results are based on the performance of the algorithms run in MATLAB_2015a on a 16 GB RAM computer. In both methods, super-components are employed for a fair comparison. For the system in Figure 5.4.1.6, when n reaches 12, the storage demand for the JT algorithm exceeds memory. Results for the JT compared to the proposed algorithms as the number of super-components in the system increases is shown in Figure 5.4.1.7. Table 5.4.1.7 lists these values and computes the data compression ratio of the proposed algorithms, i.e., the ratio of the number of elements required for the "JT" compared to "Proposed" algorithms.

In Figure 5.4.1.7 and Table 5.4.1.7, the values being recorded are the maximum number of elements stored during construction of and inference over the BN for systems of increasing size. For the proposed algorithms, the value includes both the elements in compressed intermediate factors $c\lambda_k$ and the bundles in the dictionaries d_k used in defining $c\lambda_k$. From Figure 5.4.1.7, we

see that the proposed algorithms achieve significant savings in the memory storage requirement for the BN model. The maximum number of elements stored is orders of magnitude smaller than required for the JT algorithm. For this example, the memory storage in both cases increases exponentially with system size. However, the base of the increase for the proposed algorithms is around 2 compared to 5 for JT. For the 11 super-components case, which represents a system of 88 components, the proposed algorithms require 26580 elements to be stored. This is compared to $5^{88} = 3.23 \times 10^{61}$ elements required for the original formulation.

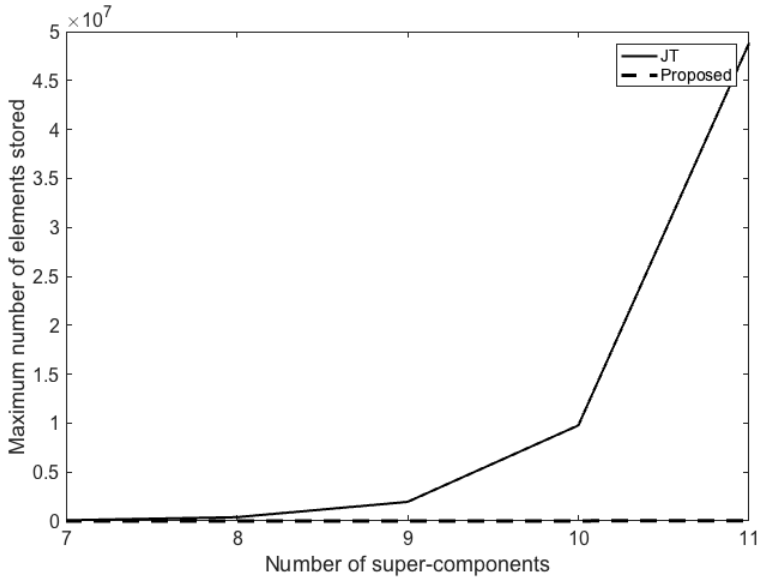

Figure 5.4.1.7 Memory storage requirements of JT compared to proposed algorithms

Table 5.4.1.7 Comparison of memory storage required for JT vs. proposed algorithms

	Number of super-components					
	7	8	9	10	11	12
JT	78125	390625	1953125	9765625	48828125	Exceeds
Proposed	2025	11080	11580	14080	26580	
Data compression ratio	38.58	35.26	168.66	693.58	1837.03	

For the JT algorithm, when the number of super-components in the system increases to 12, the size of the CPT exceeds the available memory storage capacity and the BN model cannot be constructed. Further, the last row in Table 5.4.1.7 shows the number of times by which the data has been compressed, i.e., the data compression ratio, using the proposed algorithms. We note again that the compression is lossless, so we are not losing any information nor are we making any approximations during the compression and inference processes. From the results, we see that as the size of the system increases, so does the compression efficiency of the proposed algorithms, with an exponentially increasing data compression ratio.

Algorithm performance: computational efficiency

Now we examine the performance of the proposed algorithms in terms of computational efficiency. Table 5.4.1.8 lists the computation time needed for calculation over the network for "JT" compared to "Proposed." The JT algorithm time includes constructing the full CPT and calculating the prior probability distribution $p(sys)$. The proposed algorithms time includes computation required for both compression and preprocessing information in the BN.

Table 5.4.1.8 Comparison of computation time for calculation for JT vs. proposed algorithms (unit: sec)

	Number of super-components					
	7	8	9	10	11	12
JT	24.01	131.29	710.01	3564.23	17563.11	Exceeds
Proposed	35.00	188.68	1077.10	6273.59	26519.32	
Proposed/JT	1.46	1.44	1.52	1.76	1.51	

From Table 5.4.1.8, we see that the computation times increase as the size of the system increases, a well-known example of the problem of dimensionality. The time required for the proposed algorithm is approximately 1.5 times that of the JT algorithm. However, with the large savings in memory storage, we consider the performance metrics for the proposed algorithms an acceptable tradeoff. For example, for a system of 11 super-components, the data is compressed by 1837 times while the computation time increases by 1.51 times.

Memory compared to computation time requirements are also fundamentally different. Memory storage is a hard constraint. If the maximum required memory exceeds storage capacity of a program or machine, no further analysis can be performed. While it is true that memory can be distributed, e.g., in cloud storage, a hard limit still exists. In contrast, computation time is more flexible. Indeed, methods such as parallel computing exist to address computation time. Further, for the proposed algorithms, the compression efficiency grows significantly as the number of components increases, while the computation time ratio remains stable. Thus, the algorithms enable larger multi-state flow systems to be modeled using BNs than previously possible for probabilistic inference and reliability assessment.

5.5 Contributions

In this chapter, we propose new algorithms for constructing and performing inference in BN models for reliability assessment of multi-state infrastructure flow networks. The new algorithms address the major system size limitation in the use of BNs for modeling large systems and the increased complexity of modeling multi-state flow compared to binary connectivity networks. They include a super-component substitution method, which reduces the total number of nodes

in the BN, and component renumbering heuristics to increase computational efficiency. Algorithms to losslessly compress the system CPT while simultaneously calculating and compressing intermediate factors for exact inference are then proposed. The performance of the proposed algorithms is tested using an example system. Compared with existing methods, the new algorithms achieve orders of magnitude savings in memory storage. This is accompanied by a slight decrease in computational efficiency. However, the compression efficiency improves with an exponentially increasing data compression ratio, while the computation time ratio remains stable, as the size of the system increases. Together, these algorithms enable multi-state flow networks to be modeled as BNs. As infrastructures age and are subjected to increasing hazards, the use of BNs for assessment over a variety of scenarios, including updating with new information, enables prioritization of components and decision-making across the network to increase the reliability of these critical systems.

Chapter 6 Flow capacity – modified maximum flow theory

The rest of the chapter is organized as the following. First, related work and background regarding freight network reliability analysis and Maximum Flow Theory is introduced. After that, the target network and the scope of this project are defined. A graphical solution based on Maximum Flow Theory is proposed with detailed discussion. Two major modifications are made with respect to the traditional method - extended application on multiple-source-multiple-sink scenario and additional constraints on plausible routes between different Origin-Destination (OD) pairs. A quasi-optimization process is introduced for recovery activity setup. The test applications consist of two parts. The first one is grid networks of different sizes used for comparison between the mathematical solution and graphical solution. The second one is a real-life freight transportation network - Western U.S. Double-Stack Container Network. Finally, the contributions from proposed graphical solution are concluded.

6.1 Introduction – Maximum Flow Theory

The theory of maximum flow was first proposed by Ford and Fulkerson in 1955 with the objective of finding a maximum feasible flow through a single-source-single-sink network. The Ford-Fulkerson method has two main steps: 1) Create the residual graph and 2) Find an augmenting path. A residual graph contains the potential flow increment. An augmenting path in the residual graph results in an additional flow change in the original network. The residual graph is defined as the following:

A residual graph G' of the original network G shares the same set of nodes, while for links L_{v_i,v_j}:

- A forward link L_{v_i,v_j} is created with capacity $FC_{v_i,v_j} - f_{v_i,v_j}$, if $FC_{v_i,v_j} - f_{v_i,v_j} > 0$.
- A backward link L_{v_j,v_i} is created with capacity f_{v_i,v_j}, if $f_{v_i,v_j} > 0$.

An augmenting path found in residual graph G' is then added to the flow in the original graph G. When there is no augmenting path in G', the flow in G is maximum.

An example execution of the Ford-Fulkerson method is shown for a simplified network in Figure 6.1.1 and Figure 6.1.2, where the source node is V_1 and sink node is V_6. In both Figure 6.1.1 and Figure 6.1.2, part (A) is the original graph G with link flow information X/Y listed on each link. X represents the current flow on the link and Y represents the link capacity. Part (B) is the residual graph G' with augmenting path highlighted in bold. Part (C) is the resulting graph after imposing the augmenting path in Part (B) on the previous graph in Part (A). Here, the augmenting path is separated into two cases, which will be addressed in the following discussion on extending classic maximum flow theory to the multiple-source-multiple-sink network of interest in this paper.

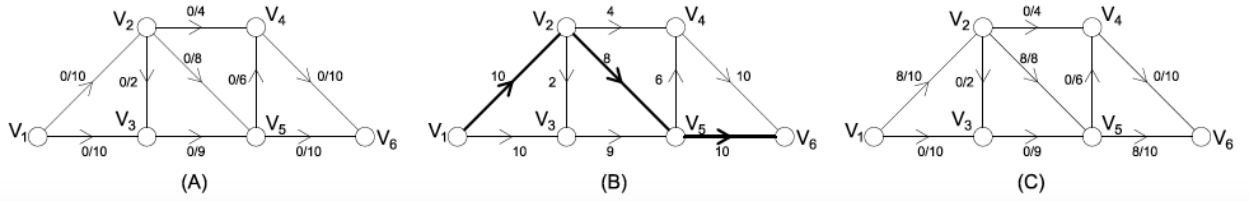

Figure 6.1.1 Augmenting path without backward links (Case 1)

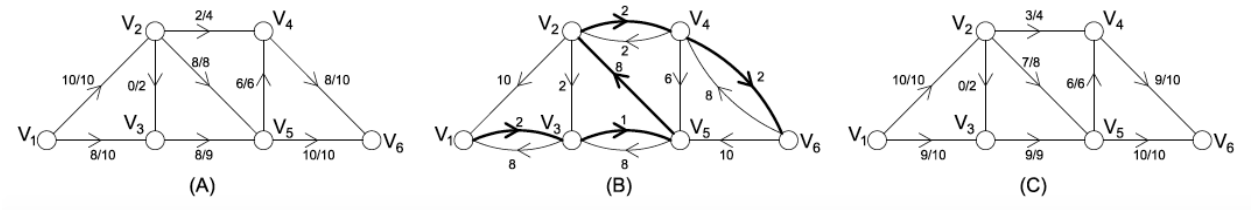

Figure 6.1.2 Augmenting path including backward links (Case 2)

In Figure 6.1.1, there are no backward links in the augmenting path in part (B). This is called Case 1 in the following discussion. The bottleneck of the augmenting path is 8. The augmenting path $\{V_1 \rightarrow V_2 \rightarrow V_5 \rightarrow V_6\}$ is the same as the flow increment path in part (C).

In Figure 6.1.2, L_{V_5,V_2} is the backward link in the augmenting path in part (B). Augmenting paths with backward links are classified as Case 2 in this paper. The bottleneck of the augmenting path $\{V_1 \rightarrow V_3 \rightarrow V_5 \rightarrow V_2 \rightarrow V_4 \rightarrow V_6\}$ is 1. To address the Case 2 scenario, the flow increment from part (A) to part (C) in Figure 6.1.2 is found by a proposed three-step procedure as shown in Figure 6.1.3. The procedure consists of a three steps – Remove, Restore, and Redirect. It is illustrated for the example graph as follows:

- Remove: 1 unit flow on path $\{V_1 \rightarrow V_2 \rightarrow V_5 \rightarrow V_6\}$ is removed in part (A) of Figure 6.1.3 resulting in part (B) of Figure 6.1.3.
- Restore: 1 unit flow on path $\{V_1 \rightarrow V_3 \rightarrow V_5 \rightarrow V_6\}$ is restored in part (B) of Figure 6.1.3 resulting in part (C) of Figure 6.1.3.
- Redirect: In part (C) of Figure 6.1.3, the augmenting path is directed as $\{V_1 \rightarrow V_2 \rightarrow V_4 \rightarrow V_6\}$, shown in bold lines. No backward link exists in the redirected augmenting path (Case 1), which is now the flow increment path in part (D).

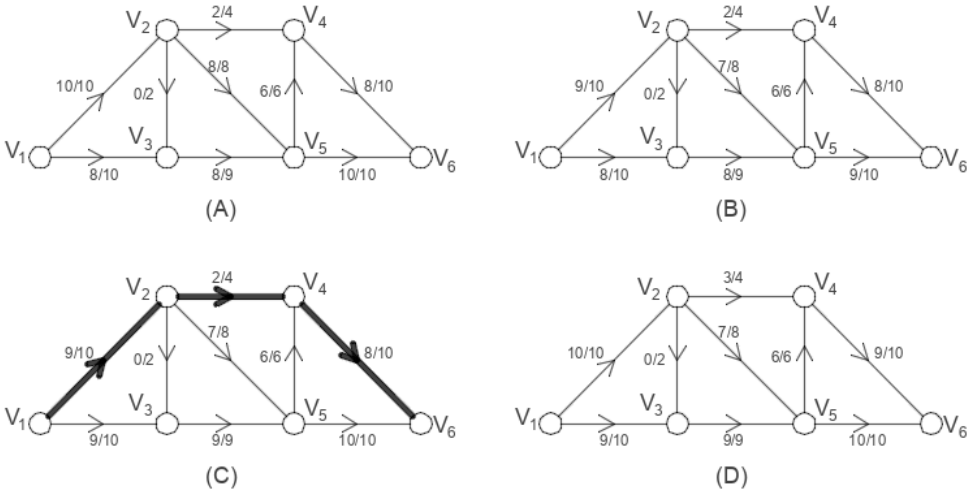

Figure 6.1.3 Three-step procedure for Case 2 augmenting path

The discussion on augmenting paths is trivial for single-source-single-sink networks. However, it is necessary for the general case of multiple sources and multiple sinks as described in the following section.

6.2 Literature review – network reliability analysis

Among the definitions of resilience, one is the capacity of a system to recover from a disturbed state to perform at or better than previous levels (Johansen et al., 2017). Resilience of a network with defined sources and sinks can be quantified in terms of demand-end accessibility (Tong and Tien 2019), or the ability for resources to be transported from sources to sinks and fulfill demands at end-point sink nodes. For the freight transport network in particular, the three parameters of interest indicating the performance of the system are time, tonnage, and cost, with differences in these parameters before and after the shock event defining the network resilience (Omer et al. 2012). However, these three parameters are not mutually independent of each other. Feasible routes are often subject to constraints on travel time and travel cost, which in return,

influence the total tonnage covered in the network. In Nair et al. (2010), tonnage resilience is considered as the only indicator for resilience assessment. Tonnage resilience is defined as the fraction of demand that can be satisfied by using specific resources while maintaining a prescribed level of service. This paper uses the tonnage resilience as a criterion for freight transport resilience.

Maximum flow algorithms, first proposed and developed by Ford and Fulkerson (2009), maximize the feasible flow through a network. In the case of a freight transport network, without loss of generality, the capacity of a link is treated as a random variable. Simulation-based methods are often used to tackle the uncertainties in system performance, i.e., Chen and Miller (2012) and Chen et al. (2017). Although an analytical solution is investigated by Han et al. (2014), it requires independency assumptions and yields an unguaranteed approximated solution. Another challenge in applying theories of maximum flow to the freight transport network is the need to address the multiple-source-to-multiple-sink scenario, where resources flow from multiple sources to multiple sinks as is realistic for the freight network. One way to tackle this challenge is to create a virtual source node and a virtual sink node, converting the problem into the single-source-single-sink case, proposed by Fang et al. (2018) for a power system. However, in a freight transport network, we are concerned with the flow between specific OD pairs instead of only the total flow. General discussion about finding maximum flows without considering the practicability of the routes are introduced by Miller and Naor (1995) and Borradaile et al. (2017). However, in practice, the limitations on feasible routes by constraints on travel time and travel cost must be considered, and cannot be accommodated with existing approaches.

In the mathematical solution approach proposed by Chen and Miller (2012), Benders decomposition, column generation, and Monte Carlo simulation are employed to solve the optimization problem to maximize the tonnage resilience over the network. Although running time in that case is measured in minutes, the path-link matrix definition needed as part of the process is computationally intractable for general networks. Rather than finding the solution to the maximum flow problem by solely solving a mathematical optimization problem, the proposed graphical approach described in this paper solves the problem under a faster computational time, increasing applicability to general flow networks. In addition, for the proposed mathematical solution in Chen and Miller (2012), the recovery activities are assumed to be implemented immediately after the event. In this paper, this assumption is relaxed. Recovery activities are assigned to different time segments through a quasi-optimal solution process, where local optimum recovery policies are used to constitute a global optimum solution.

To model link degradation and recovery, Bocchini and Frangopol (2012) use a linearly increasing curve to model the bridge restoration process. For the railway freight network discussed in this paper, a monotonically increasing step function is used as an alternative indicating stages of recovery. Alternate restoration curves can be implemented without impact on the usability of the proposed approach. For restrictions and optimization on the recovery process, González et al. (2016) specify limitations on available resources in a given geographical area. For the optimization process in this paper, similar restriction conditions on resource availability are implemented to minimize the total loss. Finally, to be able to model the link recovery in detail, the recovery process in the proposed approach is discretized into several stages, instead of relying on only binary states as in previous work (Chen and Miller, 2012). While Chen et al.

(2017) sperate the resilience analysis into two stages based on the location of the port, i.e., seaports and dry ports for a container transportation network, in this paper, transitions between different types of ports are not discussed. The focus is on creating a computationally efficient approach to solve the maximum flow problem as applied to freight networks including physical network characteristics and constraints.

6.3 Theoretical deduction

Notations

G: Freight transport network under normal conditions.

G': Residual graph of freight transport network.

G_h: Freight transport network under hazard

V_i: Node i.

L_{v_i,v_j}: Link connecting node v_i and node v_j.

f_{v_i,v_j} : The flow on link L_{v_i,v_j}.

FC_{v_i,v_j}: Flow capacity on link L_{v_i,v_j}.

T_{v_i,v_j}: Travel time on link L_{v_i,v_j}.

C_{v_i,v_j}: Travel cost on link L_{v_i,v_j}.

TC_i: Travel cost limits on the ith OD pair.

TT_i: Travel time limits on the ith OD pair.

F_i: Total flow on the ith OD pair.

N: Total number of OD pairs.

p_i^j: The jth path of the ith OD pair.

fp_i^j: Flow on p_i^j.

tp_i^j: Travel time on p_i^j.

cp_i^j: Travel cost on p_i^j.

n_i: Total number of paths connecting the ith OD pair.

O_i: Source node of the ith OD pair.

D_i: Sink node of the ith OD pair.

$\alpha(p_i^j, L_{v_i,v_j})$: binary variable on path-link incidence. 1, if $L_{v_i,v_j} \in p_i^j$; 0, otherwise.

$\mathrm{C_L}$: labor cost.

$\mathrm{C_C}$: commodity loss.

$\mathrm{S_k}$: recovery area k.

R_{S_k}: resource limit on recovery area k.

RA: all possible recovery activities.

ra: one realization of recovery activities.

$\mathrm{U}_{S_k,t}(ra)$: Utilization of resource in area $\mathrm{S_k}$ due to recovery activity ra at time t.

6.3.1 Target network

A freight transport network can be depicted as a directed graph G with nodes $\{V_i\}$ and links $\{L_{v_i,v_j}\}$. In this paper, discussion is under specified origin-destination (OD) demands and link constraints on capacity, travel time, and travel cost. The goal is to allocate commodities into the flow network satisfying the maximum OD demands. The commodity is set to be equally important, e.g., single commodity, across the network.

An example flow network consisting of 24 nodes and 32 links is shown in Figure 6.3.1.1. As an illustration of the multiple-source-to-multiple-sink scenario, nodes V_1, V_7, V_{13}, V_{19} are source nodes, and nodes $V_6, V_{12}, V_{18}, V_{24}$ are sink nodes. The value shown on the link indicates the capacity of that link. Example demands, travel time limits, and travel cost limits are listed in Table 6.3.1.1. FEUs (forty-foot equivalent units) are a measure for demand, days for travel time limits, and millions of dollars for travel cost limits. With the given notation, the optimization problem is thus defined under the following four conditions.

$$max \sum_{i=1}^{N} F_i \quad \cdots\cdots (1)$$

subject to

- Conservation law constraints:

$$F_i = \sum_{j=1}^{n_i} fp_i^j \quad \cdots\cdots (2)$$

$$f_{v_i v_j} = \sum_{i=1}^{N} \sum_{j=1}^{n_i} fp_i^j \times \alpha(p_i^j, L_{v_i,v_j}) \quad \cdots\cdots (3)$$

$$tp_i^j = \sum_{v_i v_j} T_{v_i,v_j} \times \alpha(p_i^j, L_{v_i,v_j}) \quad \cdots\cdots (4)$$

$$cp_i^j = \sum_{v_i v_j} C_{v_i,v_j} \times \alpha(p_i^j, L_{v_i,v_j}) \quad \cdots\cdots (5)$$

- Capacity constraints:

$$0 \le f_{v_i,v_j} \le FC_{v_i,v_j} \quad \cdots\cdots (6)$$

- Travel time constraints:

$$0 \leq tp_i^j \leq TT_i \quad \cdots\cdots (7)$$

- Travel cost constraints:

$$0 \leq cp_i^j \leq TC_i \quad \cdots\cdots (8)$$

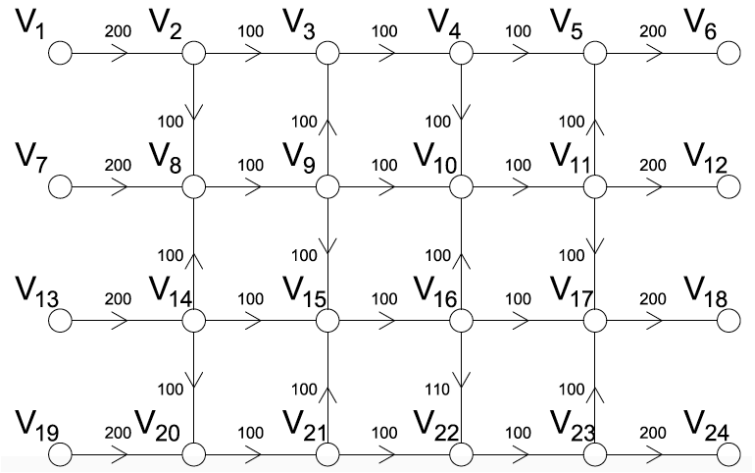

Figure 6.3.1.1 Example flow network

Table 6.3.1.1 Example constraints on OD demands

OD pair	Demand (e.g., FEUs)	Travel time limits (e.g., days)	Travel cost limits (e.g., $M)
$V_1 \rightarrow V_6$	120	8	0.8
$V_1 \rightarrow V_{18}$	150	11	1.1

Table 6.3.1.1 continued

$V_7 \rightarrow V_6$	180	9	0.9
$V_7 \rightarrow V_{24}$	24	12	1.2
$V_{13} \rightarrow V_{12}$	105	9	0.9
$V_{13} \rightarrow V_{18}$	75	8	0.8
$V_{19} \rightarrow V_{12}$	150	11	1.1
$V_{19} \rightarrow V_{24}$	115	7	0.7

Note that the proposed method to find maximum network flows is based on the theory of maximum flow, which has been developed for graphs with one-way flows. In the case of bidirectional links, the two-way edges problem (part A of Figure 6.3.1.2) can be transformed into a one-way edge problem by adding an additional node in one of the directional links (part B of Figure 6.3.1.2). In this way, the proposed method is applicable to networks with both unidirectional and bidirectional flows.

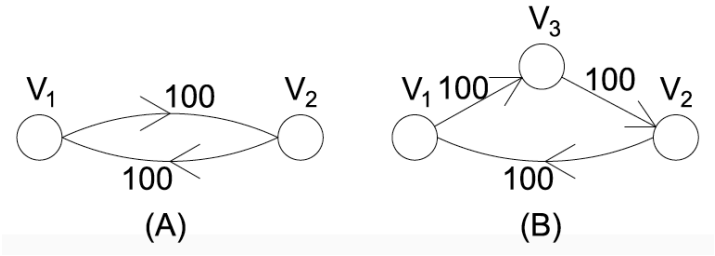

Figure 6.3.1.2 The case of bidirectional directed links

6.3.2 Modified Maximum Flow Theory

This section discusses extension of the theory of maximum flow to the multiple-source-to-multiple-sink scenario, e.g., the network shown in Figure 6.3.1.1, for freight networks of interest in this paper. Without loss of generality, the method examines all possible flow increments by searching for augmenting paths in the residual graph. As the problem includes constraints on the travel cost and travel time, all the paths that are added for flow increments are recorded for use in validating augmenting paths.

For clarity, augmenting paths are marked directly (bold lines) on the original graph as shown in Figure 6.3.2.1. Backward links contradict the flow direction of the augmenting path. For example, in part (A) of Figure 6.3.2.1, $L_{V_{14}V_{20}}$, $L_{V_9V_{15}}$, $L_{V_4V_{10}}$ (backward links) contradict the flow of the augmenting path from V_{19} to V_6. The result is an augmenting path of Case 2 as described in the previous section.

As an example, Figure 6.3.2.1 illustrates the concept of converting Case 2 augmenting paths to Case 1 augmenting paths in the proposed three-step process, i.e., the Remove-Restore-Redirect process. Consider the first-appearing backward link $L_{V_{14}V_{20}}$ in part (A) of Figure 6.3.2.1. Since all added paths have been recorded, a path that passes through the backward link $L_{V_{14}V_{20}}$ can be easily found. The authors call this a trespassing path (TP), which is shown in dashed lines in Figure 6.3.2.1. For the Remove step, a certain amount of flow, determined by the bottleneck, is removed from the trespassing path. For the Restore step, the same amount of flow is added to $\{V_{19} \rightarrow V_{20} \rightarrow V_{21} \rightarrow V_{22} \rightarrow V_{23} \rightarrow V_{24}\}$. For the Redirect step, the new augmenting path is converted as $\{V_{13} \rightarrow V_{14} \rightarrow V_{15} \rightarrow V_9 \rightarrow V_{10} \rightarrow V_4 \rightarrow V_5 \rightarrow V_6\}$, which is shown as the

augmenting path in part (B) of Figure 6.3.2.1. The process is then repeated for the first backward link in part (B) of Figure 6.3.2.1 as was previously conducted for the path in part (A). $L_{V_9V_{15}}$ is the new first-appearing backward link. A trespassing path that passes $L_{V_9V_{15}}$ is found and removed. The flows on path $\{V_{13} \rightarrow V_{14} \rightarrow V_{15} \rightarrow V_{16} \rightarrow V_{22} \rightarrow V_{23} \rightarrow V_{24}\}$ are then updated, and the augmenting path is altered to the one shown in part (C) of Figure 6.3.2.1. After applying the same methodology on the path in part (C) of Figure 6.3.2.1, the augmenting path in part (D) of Figure 6.3.2.1 no longer has any backward links, and the original Case 2 augmenting path shown in part (A) of Figure 6.3.2.1 has been successfully converted to the Case 1 augmenting path in part (D) of Figure 6.3.2.1. Note that multiple trespassing paths may exist for one backward link. Examination over all trespassing path combinations under the constraints is needed for the maximum flow assignment.

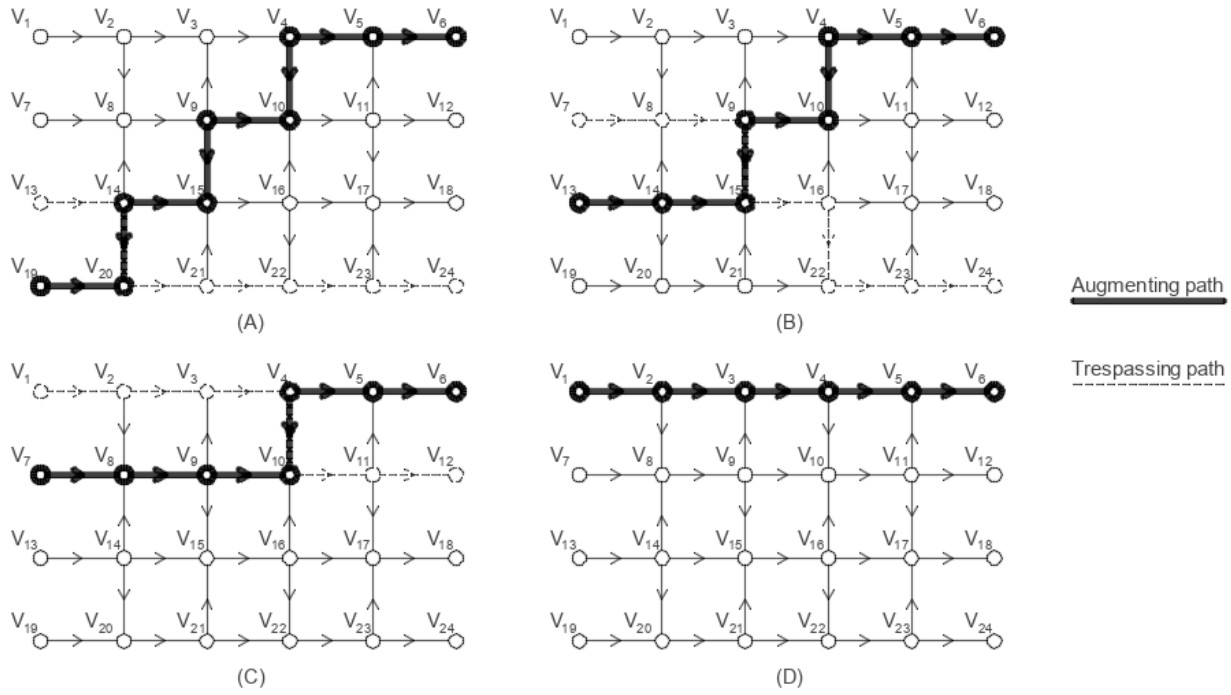

Figure 6.3.2.1 Case 2 augmenting path in multiple-source-multiple-sink networks

As described, a key to the proposed method is the converting of Case 2 to Case 1 augmenting paths. Unlike a Case 1 augmenting path (e.g., Figure 6.1.2), which has an explicit flow increment path, a Case 2 augmenting path is more complicated as it involves removing and restoring the flows on other paths. This interaction with other added paths makes is difficult to check the flow increment, especially when travel cost and travel time limits on feasible paths must be considered. However, by changing the Case 2 augmenting path into a Case 1 augmenting path, the challenge on identifying the flow change in the network can be solved, with the flow increment defined based on travel cost (Eq. 8) and travel time (Eq. 7) constraints. The methodology for finding the original residual graph and augmenting paths is according to the classic maximum flow theory. The subsequent case discussion and treatment of Case 1 and Case 2 augmenting paths is unique to the proposed approach. The full workflow of the proposed method is shown in Figure 6.3.2.2.

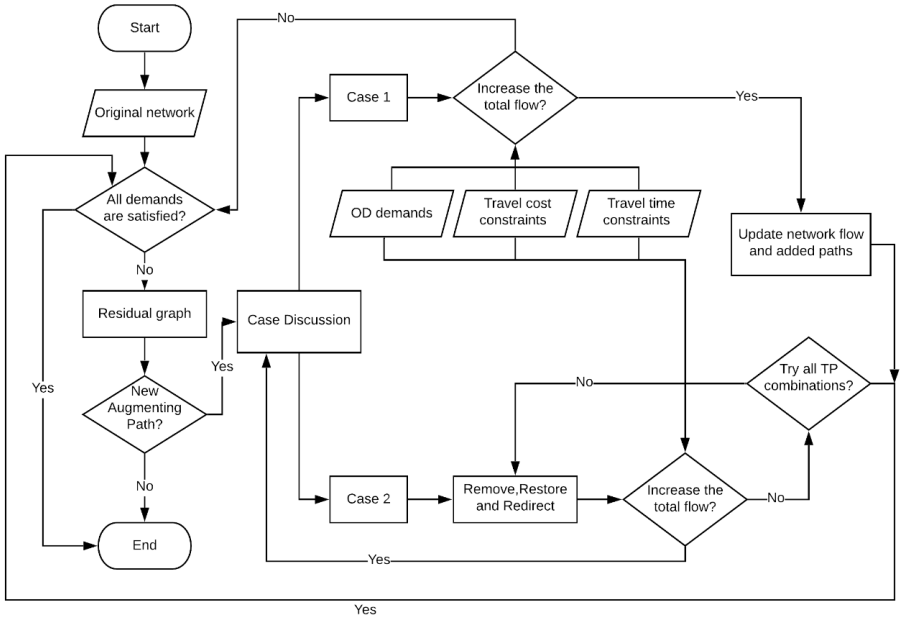

Figure 6.3.2.2 Workflow for maximizing total network flow

6.3.3 Pseudo-code for modified Maximum Flow Theory

To facilitate implementation of the proposed approach for the reader, the following pseudocode describes treatment of Case 1 and Case 2 augmenting paths, including how it is determined if the augmenting path will increase the flow.

```
Input: {Case number}, {Augmenting path}, {OD demands}, {Travel cost limits}, {Travel time
limits}, {Added paths}, {Travel cost}, {Travel time}
Output: {Decision}

For Case #1:
        {Augmenting path}→{Source node, Sink node}.
        {Source node, Sink node}, {OD demands}, {Added paths}→{Remaining demand}.
        {Source node, Sink node}, {Travel cost limits}→{Max travel cost}.
        {Source node, Sink node}, {Travel time limits}→{Max travel time}.
        {Augmenting path}, {Travel time}→{Actual travel time}.
        {Augmenting path}, {Travel cost}→{Actual travel cost}.
        If {Remaining demand} is 0 or {Actual travel time}>{Max travel time}
        or {Actual travel  cost}>{Max travel cost}:
                Return {No}.
        Else:
                Return {Yes}.
```

For Case #2:

{Added paths}→{Trespassing path}. #Pick one from all possible TPs

{Augmenting path}→{Bottleneck}.

Go through the <Remove, restore and redirect process>→{Removed path},

{Restored path} and {Redirected augmenting path}.

{Removed path}→{Source node 1, Sink node 1}.

{Restored path}→{Source node 2, Sink node 2}.

{Source node 2, Sink node 2}, {Travel cost limits}→{Max travel cost}.

{Source node 2, Sink node 2}, {Travel time limits}→{Max travel time}.

{Restored path}, {Travel time}→{Actual travel time}.

{Restored path}, {Travel cost}→{Actual travel cost}.

If {Actual travel time}>{Max travel time} or {Actual travel cost}>{Max travel cost}:

Return {No}.

Else:

If {Source node 1, Sink node 1} is same as {Source node 2, Sink node 2}:

{Removed path}, {Bottleneck}$\xrightarrow{Update}$ {Added paths}.

{Restore path}, {Bottleneck}$\xrightarrow{Update}$ {Added paths}.

{Redirected augmenting path}$\xrightarrow{Update}$ {Augmenting path}.

Return {Yes}.

Else:

{Source node 2, Sink node 2}, {OD demands}→{Remaining demands}.

{Bottleneck}=min{{Bottleneck}, {Remaining demands}}.

If {Bottleneck} is 0:

Return {No}.

Else:

{Removed path}, {Bottleneck} $\xrightarrow{Update}$ {Added paths}.

{Restore path}, {Bottleneck} $\xrightarrow{Update}$ {Added paths}.

{Redirected augmenting path} $\xrightarrow{Update}$ {Augmenting path}.

Return {Yes}.

6.3.4 Quasi-optimization process

The workflow shown in Figure 6.3.2.2 provides an approach to calculate maximum values of total network flow. In disruption scenarios, maximum values will change. This section describes the quasi-optimization process to evaluate system performance under disruption scenarios and to recommend a strategy for network recovery and resilience assessment.

In a disruption, the commodity loss ($\mathrm{C_C}$) is estimated by the difference between the current maximum flow and maximum flow under normal conditions. Its value over time is a function of the recovery policy selection, network under hazard, and network under normal conditions, i.e., $f_1(ra, G_h, \mathrm{G})$. The recovery process formulation also includes the labor cost ($\mathrm{C_L}$), which is a function of recovery activity choice, i.e., $f_2(ra)$. Finally, a constraint on the recovery activity based on available resources is implemented, with utilization of resources in area $\mathrm{S_k}$ due to recovery activity ra at a time t ($\mathrm{U}_{S_k,t}(ra)$) limited by the available resource $\mathrm{R}_{\mathrm{S}_k}$ in that area $\mathrm{S_k}$. The optimization problem is given in equation (9). The objective is to minimize the summation of labor cost and commodity loss.

$$\text{Min } \mathrm{C_L} + \mathrm{C_C} \quad \cdots\cdots (9)$$

$$\text{s.t. } \mathrm{C_L} = f_1(ra, G_h, \mathrm{G}), ra \in RA \quad \cdots\cdots (10)$$

$$\mathrm{C_c} = f_2(ra), ra \in RA \quad \cdots\cdots (11)$$

$$\mathrm{U}_{S_k,t}(ra) \leq \mathrm{R}_{\mathrm{S}_k} \; \forall \mathrm{k}, \mathrm{t} \quad \cdots\cdots (12)$$

For the sake of computational efficiency, to solve the optimization problem, the recovery activity is optimized sequentially by regions, as in quasi-optimal solutions provided by González et al. (2016). Each region is treated separately and independently, with the assumption that the strategy picked by one region will not affect the optimal strategy picking in another region. Although locally optimal solutions are chosen for all regions, the combination of all local optima does not guarantee a global optimum. Thus, the recovery policies given in this paper are quasi-optimal. The policies are chosen from a selection of available recovery strategies made by stakeholders and decision makers with, for example, given recovery times, resource use values, and costs. The workflow for the quasi-optimization process is shown in Figure 6.3.4.1. The block labeled modified maximum flow theory indicates implementation of the proposed process to find maximum flow values for multiple-source-to-multiple-sink networks as described in the previous section.

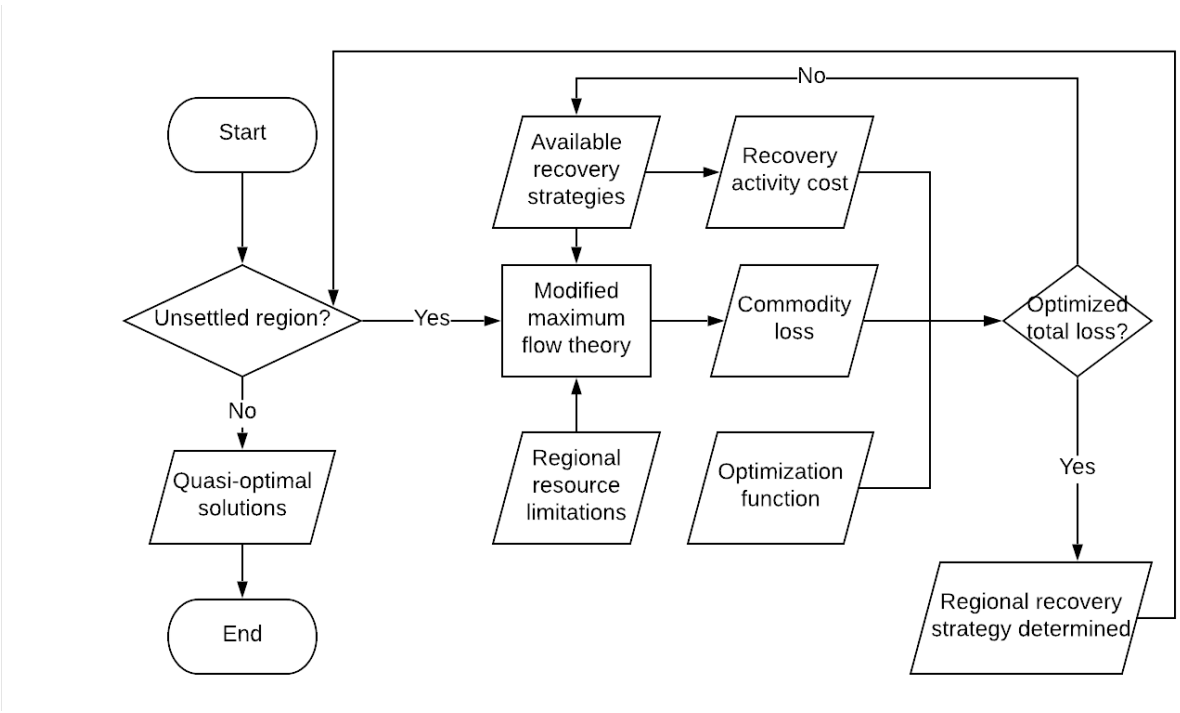

Figure 6.3.4.1 Quasi-optimal solutions finding scheme

6.5 Test application – Grid network, Western U.S. double-stack container network

This section describes application of the proposed method on two test networks. Analyses of both test applications are completed on a 16 GB RAM computer in MATLAB_R2017b. The first test network is a general grid network of increasing size, enabling comparison between the prior mathematical solution and the proposed graphical solution. It is shown that the path-link matrix, which requires high computational demands, is not needed in the proposed graphical solution, resulting in significant savings in computational cost. In the second application, a dynamic restoration process is presented in a real-world Western U.S. Double Stack Container Network under a hazard scenario. The link recovery process is discretized into 10% maximum capacity steps. The system recovery curve is obtained through the quasi-optimal solutions finding scheme.

6.5.1 Grid network

In this test application, performance between the mathematical solution, which includes building the path-link matrix for the network, and the proposed graphical solution approach are compared. Three two-way grids are analyzed, sizing from 3X3 to 5X5 as shown in Figure 6.5.1.1. For each grid, the top left and bottom left nodes serve as source nodes marked S; the top right and bottom right nodes serve as terminal sink nodes marked T. For simplicity and with the goal of comparing computational performance of the proposed compared to prior mathematical solution, no travel cost and travel time constraints are implemented. Each link has a capacity of 20 units. 10 units are expected to be shipped from each source node to sink node.

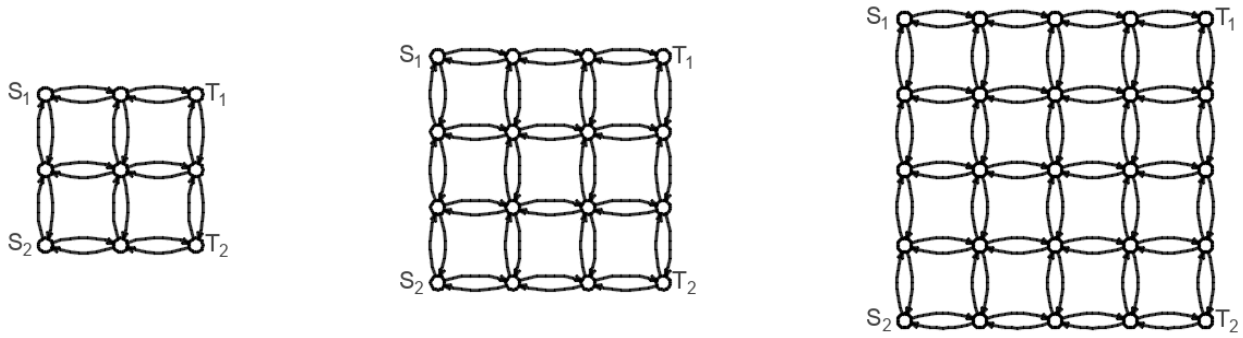

Figure 6.5.1.1 Tested grid network of different sizes

To define the path-link matrix, i.e., to find a path between a source node and a sink node, the standard depth-first search is used. Table 6.5.1.1 provides results showing the computational efficiency for the two approaches. For the mathematical solution, the computational cost for only the path-link matrix construction step is included. As this is the most computationally burdensome step of finding a solution, and results show the significant computational cost associated with this step on its own, the costs of other steps in the process need not be considered. As shown in Table 6.5.1.1, when the grid size increases to 5X5, defining all possible

paths between a source node and a sink node becomes intractable. In comparison, for the proposed graphical solution, there is no need to find all the connections between source and sink nodes. While the number of augmenting paths that need to be checked, and the associated computational time, increases as the grid network size increases, the proposed graphical solution shows orders of magnitude savings in computational cost compared with the prior mathematical solution approach. In addition, the computational savings increase as network size increases.

Table 6.5.1.1 Computational cost comparison between mathematical solution and proposed graphical solution

Grid size	3X3	4X4	5X5
Path-link matrix construction			
# of possible paths between a source node and a sink node	46	724	34204
Computational Time (sec)	0.5	73	97 hours
Proposed graphical solution			
# of checks of augmenting paths	10	84	710
Computational Time (sec)	0.1	1.6	27

6.5.2 Western U.S. Double Stack Container Network

This section illustrates performance of the proposed approach for a real-world railway freight transport network. The Western U.S. Double Stack Container Network previously examined in the literature and in Chen and Miller (2012) is characterized by 7 city nodes and 24 links

connecting them, as shown in part (A) of Figure 6.5.2.1. Since the proposed modified maximum flow theory does not directly apply for networks with bidirectional links, additional nodes are added as in Figure 6.3.1.2 to expand the network into part (B) in Figure 6.5.2.1 without affecting performance of the original network. The demands on the network are shown in Table 6.5.2.1 as given in Sun et al. (2006). The values can be easily updated to new demands based on available data. The Google Maps API is used to estimate the shortest time to cover each link and define the limitations on travel cost and travel time per link. For an OD pair, routes exceeding 50% of the shortest travel time are not considered. Per prior network definitions, the capacities of all links are assumed to be 36000 FEUs/month, where FEU indicates forty-foot equivalent units.

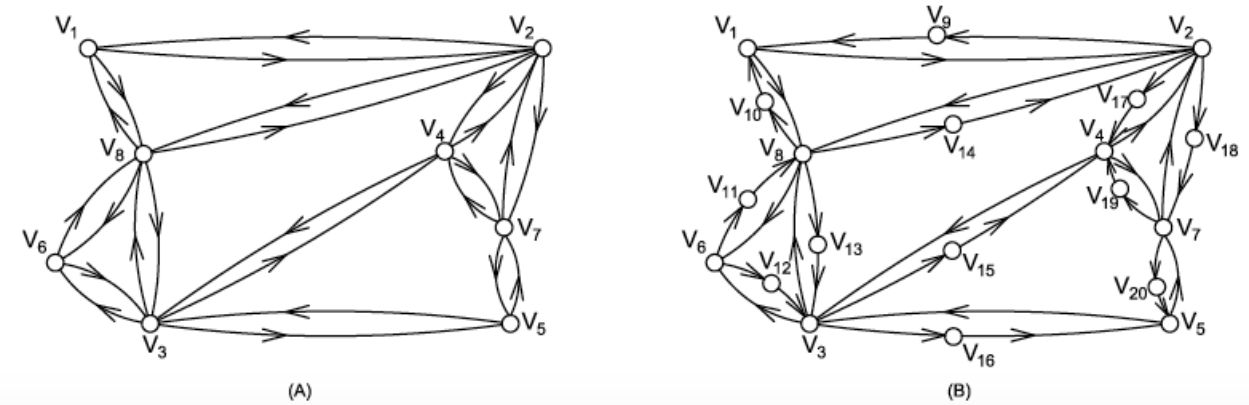

Figure 6.5.2.1 Western U.S. Double Stack Container Network

Table 6.5.2.1 OD pair demands

From	To	Volume (FEUs/month)
V_1(Seattle)	V_2(Chicago)	27057
V_2(Chicago)	V_1(Seattle)	27926
V_2(Chicago)	V_3(Los Angeles)	51030

Table 6.5.2.1 continued

V_2(Chicago)	V_4(Kansas City)	693
V_2(Chicago)	V_6(Oakland)	20167
V_2(Chicago)	V_7(Dallas)	4537
V_3(Los Angeles)	V_2(Chicago)	75577
V_3(Los Angeles)	V_4(Kansas City)	9277
V_3(Los Angeles)	V_5(Houston)	11370
V_3(Los Angeles)	V_6(Oakland)	25847
V_4(Kansas City)	V_2(Chicago)	400
V_4(Kansas City)	V_3(Los Angeles)	6067
V_5(Houston)	V_3(Los Angeles)	14223
V_5(Houston)	V_6(Oakland)	1140
V_6(Oakland)	V_2(Chicago)	11003
V_7(Dallas)	V_3(Los Angeles)	16437
V_7(Dallas)	V_5(Houston)	1373

Under normal operating conditions, when all links are working at 100% capacity, the reliability level, i.e., the ability to satisfy all network demands, reaches 94.04%. Next, a hazard scenario is considered, where an earthquake strikes V_3, and all links connected to node V_3 are influenced with varying decreases in capacity as shown in Table 6.5.2.2. For clarity, a link from 3 to 6 as listed in the table indicates the link from V_3 to V_6, and so on through the table. The recovery region is divided into three geographical areas, each with a limitation on available resources. An

example of resource availability limits is the total number of crew members available. In this case, the units for S_1 to S_3 will be number of people.

$$S_1 = \{L_{V_3,V_6}, L_{V_6,V_3}, L_{V_3,V_8}\}\ with\ resource\ limit = 240\ units$$
$$S_2 = \{L_{V_8,V_3}, L_{V_3,V_4}, L_{V_4,V_3}\}\ with\ resource\ limit = 240\ units$$
$$S_3 = \{L_{V_3,V_5}, L_{V_5,V_3}\}\ with\ resource\ limit = 160\ units$$

Table 6.5.2.2 Link capacity damage after earthquake

Link		% decrease in capacity
From	To	
3	6	40
6	3	40
3	8	30
8	3	30
4	3	50
3	4	50
5	3	20
3	5	20

For recovery activities on each link, suppose four options are available after the earthquake event. Each recovery strategy will restore 10% link capacity over a given amount of time and require certain resource units and costs as listed in Table 6.5.2.3. In this case, the cost for each recovery strategy is measured in equivalent FEUs. Other measures of cost and resources needed

can be readily implemented. The change in link capacity with respect to time during the recovery process is modeled as a monotonically increasing step function.

Table 6.5.2.3 Available recovery strategies

	Recovery period (days)	Resource (units)	Cost (FEUs)
Recovery of 10% link capacity	30	40	180000
	25	50	225000
	20	60	240000
	10	120	240000

The network recovery strategy is optimized following the process shown in Figure 6.3.4.1. The resulting recommended actions are listed in the optimized recovery schedule shown in Table 6.5.2.4. The optimized start and end days to recover each link in increments of 10% are shown. The full optimization process takes around 5 minutes of computation time to complete on a personal computer. Each run of the modified maximum flow theory to calculate maximum flow values over the freight network takes around 8 seconds.

Table 6.5.2.4 Link recovery schedule

Link		Link capacity recovery schedule (days)									
From	To	50%-60%		60%-70%		70%-80%		80%-90%		90%-100%	
		Start	End	Start	End	Start	End	Start	End	Start	End
3	6	-	-	1	10	11	30	31	60	61	90
6	3	-	-	1	30	31	40	41	60	61	90

Table 6.5.2.4 continued

3	8	-	-	-	-	1	31	31	60	61	90
8	3	-	-	-	-	1	10	11	30	31	60
4	3	1	30	31	40	41	60	61	90	61	90
3	4	1	30	31	60	61	70	71	90	91	120
5	3	-	-	-	-	-	-	1	21	21	30
3	5	-	-	-	-	-	-	1	30	31	60

The network resilience curve resulting from implementing the optimum recovery strategy is shown in Figure 6.5.2.2. Shown is the increase in fraction of demand satisfied across the network over the recovery process. Note that at the end of the recovery activity, the percent of demand met reaches 94.04%, which is the reliability level under normal link conditions. The results shown are for a network with given OD pair demands, link capacities, hazard impacts, regional resource limits, and resource costs for recovery strategies. Results shown are to illustrate the outcomes possible from applying the proposed approach. Similar analyses can be run varying these parameters to assess freight network resilience under varying scenarios.

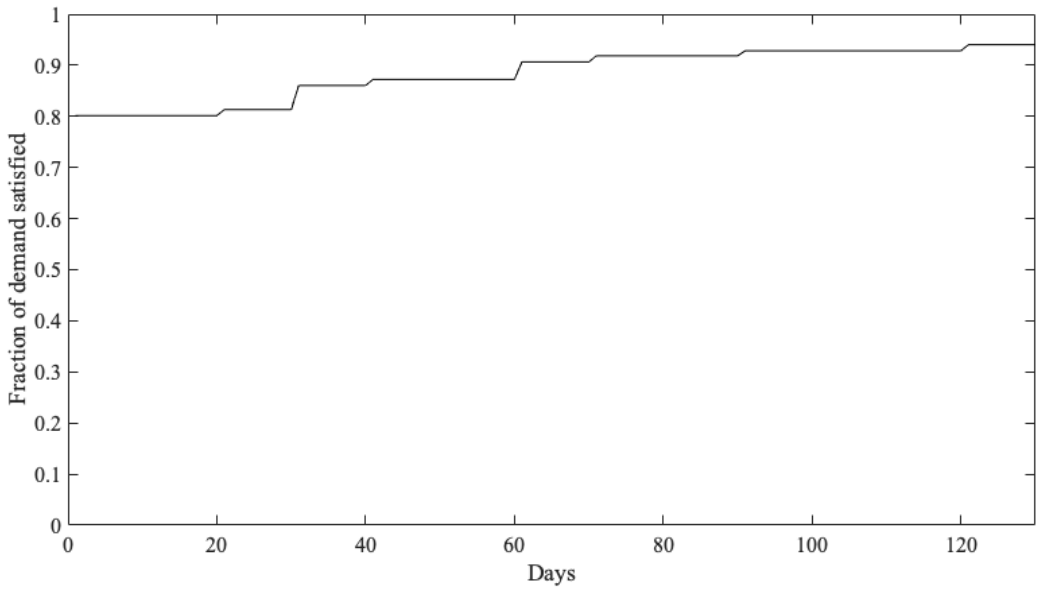

Figure 6.5.2.2 Network performance curve over the recovery process

6.3 Contributions

This chapter proposes a new method for calculating maximum flows over network graphs. The method takes a graphical approach, extending classic maximum flow theory to the case of multiple-source-to-multiple-sink networks through specifications of augmenting paths. The method is presented within a framework to quantify the resilience of freight networks, where resilience is defined as the proportion of total demands being satisfied over time after a disruption. The method enables the resilience of the network to be measured dynamically, tracking the influence of the recovery process over time. Rather than binary recovery states, recovery processes are discretized into separate stages to model the recovery policy in detail. A quasi-optimal solution for recommendation of recovery strategies takes the global optimum solution as the combination of local optimum solutions. The core modification on maximum flow theory facilitates application of the approach to the analysis of real-world networks, with

the ability to analyze networks with multiple sources and multiple sinks. Additionally, constraints on travel time and travel cost are implemented to limit the feasible routes between an OD pair. Finally, compared to the mathematical solution, the graphical solution approach does not need to predefine the path-link matrix, reducing the computational cost by orders of magnitude compared to prior approaches.

Chapter 7 Contributions and future work

7.1 RNN network

In chapter 2, we propose a modified GRU network, called Pairwise-GRU, to enhance both the accuracy and uncertainty of the nodal time series prediction. The increase in performance is achieved by adding an additional connection box between two separate GRU networks as shown in Figure 2.3.1.1. Potential improvements can be obtained by defining the neighbors and investigating modifications on the connection box between neighbors.

7.1.1 Contributions

Pairwise-GRU network is introduced in chapter 2. Comparing to the traditional GRU, our proposed Pairwise-GRU includes the influence from neighboring nodes by adding an additional hidden box between two separated single GRU network. Test performance shows that both the accuracy and confidence level of the prediction improved slightly with no significant increment on computational efficiency. The performance of the prediction improves significantly when the neighboring nodes has up-to-date information.

7.1.2 The definition of neighbors

In section 2.3.3, neighbors in the proposed Pairwise-GRU network are defined by a similarity assessment. The definition of neighbors is important because it directly influences the input data and the expected results. There are many ways to define the neighbor, e.g., physical connection, distance from the node of interest, sharing the same parent node, etc. We take a data-driven

approach to selecting the neighbor based on the data similarity between nodes. But it is by no means the only way to find the neighbor.

Also, for the sake of computational efficiency, we only consider one neighboring node to facilitate the performance of the time series prediction. For improved accuracy and uncertainty results, connecting the node with multiple neighbors can be a solution. In a situation where all nodes are connected in the network, we can update the prediction for all nodes given any newly input information.

7.1.3 The connection between neighbors

The additional connection box between two neighbors has two operators, which are picked based on the gate definition and structure in the original GRU network. We want to be consistent with the gate performance in the original GRU network. However, if we have more information about the underlying relationship between two neighbors, different operators can be used to reflect the connection. In chapter 2, the parameter learning is by trial and error on the loss function. If the structure of the connection box is changed, a potential faster computational efficiency can be found.

7.2 Probability propagation method (PrPm) and directed probability propagation method (dPrPm)

In chapter 3 and 4, we proposed PrPm and dPrPm for reliability analysis focusing on connectivity of a network. PrPm works for general complex networks with multiple sources and one sink. The solution given by PrPm is an approximated analytical solution. For dPrPm, the

approach is applicable for acyclic directed networks with multiple sources and multiple sinks. The results given by dPrPm are are upper and lower bounds of reliability. Both PrPm and dPrPm originate from the idea of belief propagation in graph theory. There are some future works that are applicable for both methods.

7.2.1 Contributions

Probability propagation method (PrPm) is proposed to analyze the reliability of general networks. Computational complexity with increasing nodes in the network n is reduced from an exponential increase $O(2^n)$ to a quartic increase $O(n^4)$. Many sampling-based approaches are limited by computational tractability to analyze rare events. For PrPm, as the method calculates the network reliability analytically, it is equally computationally efficient across reliability values.

For applications on directed acyclic network, dPrPm is proposed to further enhance the computational efficiency and accuracy. The method is applicable to the multiple-sources-multiple-sinks problem. In dPrPm, as the message contains the marginal node reliabilities, the results of dPrPm include the reliabilities of all sink nodes. dPrPm reduces computational complexity is reduced from an exponential increase $O(2^n)$ with system size to a polynomial increase $O(mn)$. Results given by dPrPm are exact bounds with100% confidence level guaranteed by dPrPm. While many sampling-based approaches are limited in the ability to analyze rare events or by computational efficiency if the probabilities of rare events are of interest, dPrPm is an analytical method and its efficiency is independent of link or network reliabilities.

7.2.2 Including more nodes in propagation

The message passed through the network in both methods are pairwise nodal distributions. During the propagation and updating process, we use the pairwise nodal distribution to estimate the three/four nodal distribution by making certain assumptions on the connections between nodes. It is foreseeable that by including more nodes in the message passing process, a more accurate result will be achieved. We pick the pairwise nodal distribution for the sake of computational efficiency. In section 3.4.3, as shown in Table 3.4.3.2, the computational time grows exponentially as we consider more nodes in propagation.

7.2.3 Diminish the uncertainty

The current updating rules are approximated analytical solutions given by assumptions made on the connections between nodes. Additional efforts can be made in the area of diminishing the uncertainty by making assumptions to more closely reflect the actual situation for a network, for example, in consideration of the network characteristics. This approach requires a graphical analysis prior to the propagation. Also, we can combine the results from PrPm and dPrPm with other methods such as the recursive decomposition algorithm (RDA) approach. The subgraph created by RDA is also applicable to PrPm and dPrPm.

7.2.4 Propagation sequence

The propagation sequence plays an important role in the performance of PrPm and dPrPm. In PrPm, at each propagation step, we pass the message to all the neighboring nodes. It impacts the

final result because in the updating rules, we make assumptions on the connections between nodes. Passing the message to all neighboring nodes at the same time saves computational time but influences the performance by the assumptions. In section 4.4.3, we run different propagation sequences 100 times to confine the gap between upper bound and low bound given by dPrPm. 100 sequences do not exhaust all possible propagation sequences. It remains a challenge on how to choose the appropriate propagation sequence.

7.3 Proposed methods on flow capacity - multistate Bayesian network and modified maximum flow theory

For the work with multistate Bayesian networks (BNs), in chapter 5, we proposed a compression algorithm and corresponding calculations based on the variable elimination inference method. Based on the performance on test application in section 5.4.1, although the storage requirement is significantly reduced, computational time is increased as a result of the preprocessing and compression steps. Thus, computational efficiency improvement is one potential future work direction.

In chapter 6, we solve the maximum flow capacity problem by using a graphical approach and a modified theory of maximum flow. We analyze the augmenting paths in tradition maximum flow theory – dividing them into two cases – and put additional limitations on feasible routes. Based on the performance of the proposed method, we can optimize the recovery strategy by comparing locally optimum solutions.

7.3.1 Contributions

In chapter 5, we propose new algorithms for constructing and performing inference in BN models for reliability assessment of multi-state infrastructure flow networks. The new algorithms address the major system size limitation in the use of BNs for modeling large systems and the increased complexity of modeling multi-state flow compared to binary connectivity networks. A lossless compression algorithm is proposed to compress the system CPT while simultaneously calculating and compressing intermediate factors for exact inference. Compared with existing methods, the new algorithms achieve orders of magnitude savings in memory storage. This is accompanied by a slight decrease in computational efficiency.

A modified maximum flow theory is proposed in chapter 6 to measure the resilience of freight network in terms of tonnage resilience. The method takes a graphical approach, extending classic maximum flow theory to the case of multiple-source-to-multiple-sink networks through specifications of augmenting paths. The method enables the resilience of the network to be measured dynamically, tracking the influence of the recovery process over time. Rather than binary recovery states, recovery processes are discretized into separate stages to model the recovery policy in detail. Additionally, constraints on travel time and travel cost are implemented to limit the feasible routes between an OD pair. Comparing to the mathematical solution, the graphical solution approach does not need to predefine the path-link matrix, reducing the computational cost by orders of magnitude compared to prior approaches

7.3.2 Computational efficiency improvement

The total computational time in the proposed multistate BN approach include two parts: compression and the calculation based on the compression result. Since most of the time is spent

on the compression, improvement on computational efficiency is likely to be found in the compression algorithm. In the compression algorithm, we use the idea of bundles to compress the total information – conditional probability tables (CPTs) – from the top to the bottom. The content in a CPT is largely influenced by the numbering of the nodes. In chapter 5, we number the nodes based on observation and their appearances in MLSs. However, there is no proof that this numbering is optimal in terms of compression rate or computational efficiency. Also, the bundle definition influences the efficiency of compression. In chapter 5, we fix the length of a bundle to the number of states, which is not a necessity for lossless compression. Reduced time on compression can be achieved by relaxing the limitations on the definition of a bundle. As a tradeoff, extra time on the calculation processes after compression will be added.

Similarly, for the proposed modified theory of maximum flow, computational efficiency improvement can be done by investigating the augmenting paths. In section 6.3.2, we divide the augmenting paths into two cases. The processing time for the case 1 scenario is straightforward and fast. For the case 2 scenario, we need to go through all added paths, which contributes to the bulk of the computational time. It will be helpful to improve the computational efficiency if the search for feasible routes prioritizes the case 1 scenario.

7.3.3 Extended application of proposed method

The application of the multistate BN approach in chapter 5 is limited to independent cases. However, the discussion can be extended to dependent cases. For the calculation part in the multistate BN modeling and analysis, we use the variable elimination method. Based on the

ordering of nodes, the calculation on the compression results can be adjusted accordingly to reflect the parental relationship between nodes, including for dependent nodes.

In the optimization discussion in section 6.3.4, we find the quasi-optimal solution for the whole system by combining the local optimum solutions. Future work can be done by investigating the global optimum solution. Also, the discussion in chapter 6 is limited to single commodity networks. In reality, in a freight network, different types of commodities are transported at the same time. Another future research direction is to extend the application to multi-type commodities.

7.4 Future applications on general networks

In this thesis, we separate the reliability assessment into two standards: connectivity and flow capacity. For a more comprehensive evaluation of the reliability of infrastructure flow network, future efforts can be spent on merging the prior two standards into one single algorithm. Here, we need to run two separate algorithms to get the connectivity and flow capacity of the infrastructure flow network. However, we find the possibility in using one algorithm to accomplish the two objects (connectivity/flow capacity analysis). The PrPm/dPrPm solve the connectivity issue by propagating the information through the neighboring node, which solves the problem graphically. Likewise, in the modified maximum flow theory, we also put forward a graphical solution to extend the traditional maximum flow theory into the discussions under real-life scenarios. Both approaches rely on graphical solution. In that sense, it is a potential future work direction to combine the algorithms for two standards into a single comprehensive

algorithm, which is expected to improve the computational efficiency, saving the computational time, while providing accurate results.

References

1. Bai, Y., Wang, P., Li, C., Xie, J., and Wang, Y., "A multi-scale relevance vector regression approach for daily urban water demand forecasting", Journal of Hydrology, No. 517, pp. 236-245, 2014

2. Barber D., Bayesian reasoning and machine learning, Cambridge University Press, 2012.

3. Benaddy M, Wakrim M., "Cutset enumerating and network reliability computing by a new recursive algorithm and inclusion exclusion principle," International Journal of Computer Applications, Vol. 45, No. 16, pp. 22–25, 2012.

4. Bensi, M., Der Kiureghian, A., and Straub, D., "Efficient Bayesian network modeling of systems," Reliability Engineering and System Safety, Vol. 112, pp. 200-213, 2013.

5. Birolini A., Reliability Engineering: Theory and Practice, 4th ed., Berlin: Springer, 2004.

6. Bobbio, A., Portinale, L, Minichino, M., and Ciancamerla, E., "Improving the analysis of dependable systems by mapping fault trees into Bayesian networks," Reliability Engineering and System Safety, Vol. 71, No. 3, pp. 249-260, 2001.

7. Bocchini, P., and Frangopol, D.M., "Restoration of bridge networks after an earthquake: Multicriteria intervention optimization", Earthquake Spectra, Vol. 28, No. 2, pp. 426-455, 2012.

8. Borradaile, G., Klein, P., Mozes, S., Nussbaum, Y., and Wulff-Nilsen, C., “Multiple-source multiple-sink maximum flow in directed planar graphs in near-linear time”, SIAM Journal on Computing, Vol. 46, No. 4, pp. 1280-1303, 2017.

9. Boudali, H., and Dugan, J. B., “A discrete-time Bayesian network reliability modeling and analysis framework,” Reliability Engineering and System Safety, Vol. 87, pp. 337-349, 2005.

10. Bouissou, M., and Pourret, O., “A Bayesian Belief Network Based Method for Performance Evaluation and Troubleshooting of Multistate Systems,” International Journal of Reliability, Quality and Safety Engineering, Vol. 10, No. 4, pp. 407-416, December 2003.

11. Bulteau S., and El Khadiri M., "A Monte Carlo simulation of the flow network reliability using importance and stratified sampling," PhD diss., INRIA, Vol. 32, pp. 271-287, 1998.

12. Bureau of Transportation Statistics. <https://www.bts.gov/topics/freight-transportation/freight-shipments-mode>, accessed September 9, 2019.

13. Chen, H., Cullinane, K., and Liu, N., “Developing a model for measuring the resilience of a port-hinterland container transportation network”, Transportation Research Part E, Vol. 97, pp. 282-301, 2017.

14. Chen, L., and Miller, E., “Resilience: An Indicator of Recovery Capability in Intermodal Freight Transport”, Transport science, Vol. 46, No. 1, pp. 109-123, February, 2012.

15. Cheng L., Lu Z., Zhang L., “Application of Rejection Sampling based methodology to variance based parametric sensitivity analysis,” Reliability Engineering and System Safety, Vol. 142, pp. 9-18, 2015

16. Cheng W., Cox J., Whitlock P., “Random walks on graphs and Monte Carlo methods,” Mathematics and Computers in Simulation, Vol. 135, pp. 86-94, 2017.

17. Coughlan J., A Tutorial Introduction to Belief Propagation, The Smith-Kettlewell Eye Research Institute, Aug 2009.

18. Cho, K., Merrienboer, B., Culcehre, C., Bahdanau, D., Bougares, F., Schwenk, H., and Bengio., Y., “Learning phrase representations using RNN encoder-decoder for statistical machine translation”, arXiv preprint arXiv:1406.1708, pp. 1724-1734, 2014.

19. Dechter, R., “Bucket Elimination: a Unifying Framework for Reasoning,” Artificial Intelligence, Vol. 113, pp. 41-85, 1999.

20. Deng, J., “Control problems of grey systems”, Systems and control letters, Vol. 1, No. 5, pp. 288-294, 1982.

21. Der Kiureghian A., Song J., “Multi-scale reliability analysis and updating of complex systems by use of linear programming,” Reliability Engineering and System Safety, Vol. 93, pp. 288-297, 2008

22. Dotson, W., Gobien, J.O., “A New Analysis Technique for Probabilistic Graphs,” IEEE Transactions on Circuits and Systems, Vol. 26, No. 10, pp. 855–865, 1979.

23. Dueñas-Osorio, L., “Reliability of Grid Networks and Recursive Decomposition Algorithms,” <http://duenas-osorio.rice.edu/Content.aspx?id=2147483674>, accessed April 2017.

24. Ebeling C.E., An introduction to Reliability and Maintainability Engineering 2nd. Waveland press, Inc. pp.110-112, 2010.

25. Fang, J., Su, C., Chen, Z., Sun, H., and Lund, P., “Power System Structural Vulnerability Assessment Based on an Improved Maximum Flow Approach”, IEEE Transaction on smart grid, Vol. 9, No. 2, March 2018.

26. Ford, L.R., and Fulkerson, D.R., “Maximal flow through a network”, Classic papers in combinatorics, Birkhäuser Boston, pp. 243-248, 2009.

27. González, A.D., Dueñas-Osorio, L., Sánchez-Silva, M., and Medaglia, A. L., "The interdependent network design problem for optimal infrastructure system restoration", Computer-Aided Civil and Infrastructure Engineering, Vol. 31, No. 5, pp. 334-350, 2016.

28. Gu, Y.K, and Yang, Z.X., "Reliability Analysis of Multi-State Systems Based on Bayesian Network," 2013 International Conference on Quality, Reliability, Risk, Maintenance, and Safety Engineering (QR2MSE), pp. 332-336, 2013.

29. Han, S., Peng, Z., and Wang, S., "The maximum flow problem of uncertain network", Information Sciences, Vol. 265, pp.167-175, 2014.

30. Hochreiter, S., and Schmidhuber, J., "Long short-term memory" Neural computation, No. 9, Vol. 8, pp. 1735-1780, 1997.

31. Holzfuss, J., and Mayer-Kress, G., "An approach to error-estimation in the application of dimension algorithms", Dimensions Entropies in Chaotic Systems, pp. 114-122, 1986

32. Iglesias, G., and Wastner, W., "Analysis of Similarity Measures in Time Series Clustering for the Discovery of Building Energy Patterns", Energies, No. 6, Vol. 2, pp. 579-597, 2013.

33. Jensen, F.V., and Nielsen, T.D. Bayesian Networks and Decision Graphs, 2nd ed. New York: Springer, 2007.

34. Jiang, X., Bai, R., Atkin, J., and Kendall, G., "A scheme for determining vehicle routes based on Arc-based service network design," Information Systems and Operational Research, Vol. 55, No. 1, pp. 16-37, 2016.

35. Johansen, C., and Tien, I., "Probabilistic multi-scale modeling of interdependencies between critical infrastructure systems for resilience," Sustainable and Resilient Infrastructure, in press.

36. Johansen, C., Horney, J., and Tien, I. "Metrics for evaluating and improving community resilience," Journal of Infrastructure Systems, 2016.

37. Kayacan, E., Baris, U., and Kaynak O., "Grey system theory-based models in time series prediction", Expert systems with applications, Vol.37, pp.1784-1789, 2010.

38. Khakzad, N., Khan, F., and Amyotte, P., "Safety Analysis in Process Facilities: Comparison of Fault Tree and Bayesian Network Approaches," Reliability Engineering and System Safety, Vol. 96, pp. 925-932, 2011.

39. Kim, M. C., "Reliability Block Diagram with General Gates and its Application to System Reliability Analysis," Annals of Nuclear Energy, Vol. 38, pp. 2456-2461, 2011.

40. Kim, Y., Kang, W., “Network reliability analysis of complex systems using a non-simulation-based method,” Reliability Engineering and System Safety, Vol. 110, pp. 80-88, 2013.

41. Li J, Qian Y, Liu W., “Minimal cut-based recursive decomposition algorithm for seismic reliability evaluation of lifeline networks,” Earthquake Engineering and Engineering Vibration, Vol. 6, No. 1, pp. 21–28, 2007.

42. Li, S., Li, W., Cook, C., Zhu, C., and Gao, Y., “Independently Recurrent Neural Network (IndRNN): Building a Longer and Deeper RNN”, Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR), pp. 5457-5466, 2018.

43. Lim, H.-W., Song, J., “Efficient risk assessment of lifeline networks under spatially correlated ground motions using selective recursive decomposition algorithm,” Earthquake Engineering and Structural Dynamics, Vol. 41, No. 13, pp. 1861-1882, 2012.

44. Liu, W., and Li, J., “An improved recursive decomposition algorithm for reliability evaluation of lifeline networks”, Earthquake engineering and engineering vibration, Vol. 8, No. 3, pp. 409-419, September, 2009.

45. Mahadevan, S., Zhang R., and Smith, N., “Bayesian networks for system reliability reassessment,” Structural Safety, Vol. 23, pp. 231-251, 2001.

46. Meng, F.C., “Comparing criticality of nodes via minimal cut (path) sets for coherent systems.” Probability in the Engineering and Informational Sciences, Vol. 8, No. 1, pp. 79-87, 1994.

47. Miller, G., and Naor, J., “Flow in planar graphs with multiple sources and sinks”, SIAM Journal on Computing, Vol. 24, No. 5, pp. 1002-1017, 1995.

48. Muriel-Villegas, J.E., Alvarez-Uribe, K.C., Patino-Rodriguez, C.E., and Villegas, J.G., “Analysis of transportation networks subject to natural hazards-Insights from a Colombian case”, Reliability Engineering and System Safety, Vol. 152, pp. 151-165, August 2016.

49. Murphy, K. P., “The Bayes Net Toolbox for Matlab,” Computing Science and Statistics: Proceedings of the Interface, Vol. 33, October 2001.

50. Nair, R., Avetisyan, H., and Miller, E., “Resilience Framework for Ports and Other Intermodal Components”, Transportation Research Record, Vol. 2166, No. 1, pp. 54-65, 2010.

51. Nikolaev, N. Y., and Iba H., “Polynomial harmonic GMDH learning networks for time series modeling”, Neural Networks, Vol.16, pp. 1527-1540, 2003.

52. Omer, M., Mostashari, A., Nilchiani, R., and Mansouri, M., “A framework for assessing resiliency of maritime transportation systems”, Maritime Policy & Management, Vol. 39, No. 7, pp. 685-703, 2012.

53. Ostrom, D., “Database of seismic parameters of equipment in substations.” <http://peer.berkeley.edu/lifelines/lifelines_pre_2006/final_reports/413-FR.pdf>, 2004, accessed January 10, 2017.

54. Shelekhova, V. Y., “Harmonic algorithm GMDH for large data volume”, Systems Analysis-Modelling-Simulation, Vol.20, pp.117-126, 1995.

55. Shields, M. D., Teferra, K., Hapij, A., Daddazio, R.P., “Refined stratified sampling for efficient Monte Carlo based uncertainty quantification”, Reliability Engineering and System Safety, Vol. 142, pp. 310-325, 2015.

56. Shin, Y.Y., and Koh, J.S., “An Algorithm for Generating Minimal Cutsets of Undirected Graphs,” Korean Journal of Computational and Applied Mathematics, Vol. 5, No. 3, pp. 681-693, 1998.

57. Sogin, S., Barkan, C.P., and Saat, M.R. “Simulating the effects of higher speed passenger trains in single track freight networks”, In Proceedings of the 2011 Winter Simulation Conference, IEEE, pp. 3679-3687, December 2011.

58. Spiegelhalter, D.J., Dawid, A.P., Lauritzen, S.L. and Cowell, R.G., “Bayesian analysis in expert systems,” Statistical science, pp. 219-247, 1993.

59. Suh H, Chang CK., “Algorithms for the minimal cutsets enumeration of networks by graph search and branch addition,” Proceedings of the 25th Annual IEEE Conference on Local Computer Networks, Tampa, FL7, November 8–10, 2000. pp. 100–10.

60. Sun, Y., Turnquist, M.A., and Nozick, L.K.. "Estimating freight transportation system capacity, flexibility, and degraded-condition performance", Transportation research record, pp. 80-87, Vol. 1966, No. 1, 2006.

61. Tien, I., “Bayesian network methods for modeling and reliability assessment of infrastructure systems,” Doctoral Thesis, University of California, Berkeley, 2014.

62. Tien, I., “Bayesian network methods for modeling and reliability assessment of infrastructure systems,” In Risk and Reliability Analysis: Theory and Applications, Springer International Publishing, pp. 417-452, 2017.

63. Tien, I., and Der Kiureghian, A., “Compression algorithm for Bayesian network modeling of binary systems,” In G. Deodatis, B. Ellingwood, and D. Frangopol, eds., Safety, Reliability, Risk and Life-Cycle Performance of Structures and Infrastructures, New York: CRC Press, pp. 3075-3081, June 2013.

64. Tien, I., and Der Kiureghian, A., "Compression and inference algorithms for Bayesian network modeling of infrastructure systems," In T. Haukaas, ed., Proceedings of the 12th International Conference on Applications of Statistics and Probability in Civil Engineering, Vancouver, Canada, July 12-15, 2015.

65. Tien, I., and Der Kiureghian, A., "Algorithms for Bayesian network modeling and reliability assessment of infrastructure systems," Reliability Engineering and System Safety, Vol. 156, pp. 134-147, 2016.

66. Tien, I., and Der Kiureghian, A., "Reliability assessment of critical infrastructure using Bayesian network," Journal of Infrastructure Systems, Vol. 23, No. 4, 2017

67. Tipping, M., "Sparse Bayesian Learning and the Relevance Vector Machine", Journal of Machine Learning Research, No. 1, pp. 211-244, 2001.

68. Torres-Toledano, J. G., and Succar, L. E., "Bayesian networks for reliability analysis of complex systems," Lecture Notes in Artificial Intelligence 1484, pp. 195-206, 1998.

69. Tong, Y., and Tien, I., "Algorithms for Bayesian network modeling of multi-state infrastructure flow systems," Engineering Mechanics Institute and Probabilistic Mechanics and Reliability Conference, Nashville, TN, May 22-25, 2016.

70. Tong, Y., and Tien, I., “Algorithms for Bayesian network modeling, inference, and reliability assessment for multi-state flow networks,” Journal of Computing in Civil Engineering, Vol. 31, No. 5, 2017.

71. Tong, Y., and Tien, I., “Analytical Probability Propagation Method for Reliability Analysis of General Complex Networks,” Reliability Engineering and System Safety, Vol. 189, pp. 21-30, 2019

72. Tong, Y., and Tien, I., “Probability propagation method for reliability assessment of acyclic directed networks,” ASCE Journal of Risk and Uncertainty in Engineering Systems, Part A: Civil Engineering, Vol. 5, Issue 3, September 2019b.

73. Xu., N., and Zhang, X., “Traffic volume prediction based on improved grey self-adaptable prediction formula”, Proceedings of the Ninth International Conference on Machine Learning and Cybernetics, Qingdao, pp.11-14, 2010.

74. Yedidia, J.S., Freeman, W.T., Weiss, Y., “Understanding belief propagation and its generalizations”, Exploring artificial intelligence in the new millennium, pp.236-239, 2003.

75. Yeh, W.C. “Evaluating the Reliability of a Novel Deterioration-Effect Multi-State Flow Network,” Information Sciences, Vol. 243, pp. 75-85, 2013.

76. Ziv, J., and Lempel, A., “A Universal Algorithm for Sequential Data Compression,” IEEE Transactions on Information Theory, Vol. 23, No. 3, pp. 337-343, May 1977.

77. Zuev, K., Wu, S., Beck, J., “General network reliability problem and its efficient solution by Subset Simulation”, Probabilistic Engineering Mechanics, Vol. 40, pp.25-35, 2015.